コンピュータ、数学の問題を解き始める

そして、数学者は、
証明にコンピュータを使い始める

2022/03/26 マルレク
https://www.marulabo.net/docs/math-proof/

はじめに

小論は、二つのトピックスから構成されている。

前半の第一部・第二部では、AIに数学の問題を解かせようという、AIの分野でのOpen-AIによるごく最近の挑戦的な取り組みを紹介している。

後半の第三部・第四部では、近年一部で盛り上りをみせている数学の証明問題を解くのにコンピュータを利用しようという動きを紹介している。

はじめに

小論の前半と後半で取り上げた動きは別のものである。

前者では数学の問題を解く主体として期待されているのは、いわゆる「AI」なのだが、後者では数学者がコンピュータを利用して問題を解く。主体は数学者である。

ただ、数値計算・データ処理・モデル構築・グラフィック表示等々といった、従来の数学でのコンピュータ利用の枠を超えて、「数学の証明」とコンピュータの接点を探るという点では、両者は、共通の問題領域での動きだと考えることができる。

はじめに

両者の結びつきは、実は、深いものだと僕は考えている。

なぜなら、小論での叙述の順序は逆になっているが、前者のOpen-AI Theorem Proverのもっとも重要な訓練用データの中核をなし、定理証明サンプルのデータベースとして利用されているmathlibは、まさに、後者のムーブメントの産物だからである。

前者の取り組みの主論文、”Formal Mathematics Statement Curriculum Learning” のタイトルは、そのことをよく示していると僕は思う。

はじめに

mathlibだけではない。ここでの”Formal Mathematics Statement”は、コンピュータによる形式的証明がなされている定理群を指す。こうした「学習対象」は、後者の取り組み以前には存在しなかったものだ。

2013年のプリンストンでの”Special Year”での Awody, Coquand による “Large-Scale Formalization of Mathematics” の展望は、それまでの、ヘイルズやゴンティエらの先進的だが散発的だった証明へのコンピュータ利用の本格的な拡大を呼びかけたものだ。

こうした流れの拡大の中で、Open-AI Theorem Proverの「学習対象」が生み出されたのだ。

はじめに

"Univalent Foundations and the Large-Scale Formalization of Mathematics"

https://www.ias.edu/ideas/2013/awodey-coquand-univalent-foundations

mathlibの取り組みとその広がりについては、次のページをを参照されたい。

Lean and its Mathematical Library

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

はじめに

小論の後半部分については、いくつかの補足が必要かもしれない。残念ながら、今回のセミナーでは、それを十分に行うことはできなかった。

補足されるべきことは、環境を整備して自分のマシンで「連続体仮説」や「Feit–Thompson 定理」の「証明」を実行してみた時に感ずる「違和感」に関係している。

「僕は、証明したのだろうか？」

明らかに、「僕は、証明していない。」

そこには、デカルトの言うような「明証性」は存在していない。

はじめに

それにもかかわらず、「証明」は行われている。

「機械が論理的に推論する」とか「機械が数学的証明を行う」という言葉が腑に落ちない人は多いと思う。

ただ、僕は、コンピュータには、「論理的に推論する」能力も「数学的証明を行う」能力があると考えている。

それは少なくない人が感じている、「いつか、AIは数学的証明を行うようになるかもしれない」という漠然とした期待より、ずっと強い主張であることに留意してほしい。

はじめに

それは、理論的にも歴史的にも、裏付けできるのだ。

まず、「計算」と「証明」が、本質的には同じものであるという認識が生まれ、ついで、「プログラム」は「証明」とみなせるという認識が生まれる。

理論的には、この二つの発見で、「計算」「証明」「プログラム」という三項の関係は明確になった。

「コンピュータは、プログラムの実行という形で、証明を計算する」のである。

はじめに

こうした観点から、あらためて「機械と人間」の関係、また、「創造性」という人間の認識能力の特異性、人間の「意識」の問題を考えてみたいと思っている。

以下の資料を参照されたい。

「IT技術とCoqの世界
-- 証明 = プログラム = 計算の意味を考える」
https://www.marulabo.net/docs/coq4dev/

コンピュータ、数学の問題を解き始める

第一部　Open-AI Theorem Prover

第二部　Open-AI Theorem Proverは
何を学習したのか？

第三部　数学者、証明にコンピュータを使い始める

第四部　形式的証明の背景と変化の意味を考える

第一部

Open-AI Theorem Prover

第一部で取り上げること

第一部では、Open-AI Theorem Proverの概要を見る。

まず、Open-AI Theorem Proverがどのような問題を解いたのかをみておこう。それは、「どのようにして解いたのか？」とは別のことなのだが、最新のAI技術が、どのような問題にチャレンジしているかを、わかりやすく知ることができる。

ただ、コンピュータが解いた問題を自分が簡単に解いたからといって、「AIは、やはり、大したことができないのだ」と判断するのは早計である。同様に、自分が解けない問題をコンピュータが解いたからといって、「AIは、素晴らしい。なんでもできる。」と考えるのも、どうかと思う。

第一部のまとめ

次に、ある数学の問題が与えられた時、Open-AI Theorem Proverがどのような答えを返すのかをみていく。

Open-AI Theorem Proverは、自然言語で与えられた数学の問題を形式的な数学的な言語に翻訳する。これは、Sequence to Sequence の変換で、ニューラル・ネットワークにとっては難しいことではない。

次に、Open-AI Theorem Proverは、こうして形式化された問題を、形式的に証明しようとする。

そこで、証明支援の言語 Lean が登場する。Open-AI Theorem Proverは、Leanで形式化された問題としてLeanで問題を証明しようとする。そしてLeanでの証明が成功すれば、それを返す。

Solving (Some) Formal Math Olympiad Problems

我々は、Leanのためのニューラル定理証明システムを構築した。このシステムは、 AMC12とAIME さらには二つのIMOからの問題を含む、チャレンジングな高校生むけのさまざまな数学オリンピックの問題を解くことを学んだ。これらの問題は、標準的な数学の演習問題ではなく、アメリカから（AMC12, AIME）あるいは、全世界から（IMO）から集まった、最優秀の高校生がお互いに競争すのに使われた問題である。

この証明システムは、形式的な言明の証明を見つけるために、言語モデルを利用する。新しい証明を見つけるたびに、我々はそれを新しい訓連用データとして利用する。それは、ニューラル・ネットワークを改善し、それを繰り返すことで、はるかに難しい言明の証明の解を見つけることを可能にする。

参考資料

“Solving (Some) Formal Math Olympiad Problems”
https://openai.com/blog/formal-math/?fbclid=IwAR1y4rK76c1BMN4FOgSLl3qPXk_cXxlZmIGnsy37wDsLdadRjbMiuoK4kFE

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, Ilya Sutskever

“Formal Mathematics Statement Curriculum Learning”

https://arxiv.org/pdf/2202.01344.pdf

第一部のまとめ

イリヤ達の論文は、Deep Learning技術の到達点と課題を率直に述べたものとして、興味深いものである。この第一部で、最も重要な箇所である。以下の資料本文のテキストを参照されたい。

第一部
Open-AI Theorem Prover

Open-AI Theorem Proverが証明したこと
Open-AI Theorem Proverが行う証明
数学的証明の重要性とその難しさ
難しさにどう立ち向かうか？

Open-AI Theorem Proverが証明したこと

Open-AI Theorem Proverが、証明した問題

問題1 (AMC12 2000年第5問)

$|x-2| = p$とする。$x < 2$の時、$x - p = 2 - 2p$ を証明せよ。

証明

$x < 2$ なので、$|x-2| = -(x-2)$ 。

$p = |x-2|$を使えば、$x = 2 - p$ 。

よって、$x - p = 2 - 2p$ 。

問題2 (AMC12B 2020年第6問)

$n \geq 9$なる全ての整数について、
$((n+2)! - (n+1)!)/n!$
は、完全平方数であることを示せ。

証明

$$((n+2)! - (n+1)!)/n!$$
$$= ((n+2)(n+1)n! - (n+1)n!/n!$$
$$= (n+2)(n+1) - (n+1)$$
$$= (n+1)(n+2-1)$$
$$= (n+1)^2$$

問題3 (Math dataset より)

$f(x) = Ax + B,\ g(x) = Bx + A$とする。$A - B \neq 0$で、
$f\big(g(x)\big) - g(f(x) = B - A$ なら、$\mathrm{A} + \mathrm{B} = 0$ を証明せよ。

証明

$$f\big(g(x)\big) = A(Bx + A) + B = ABx + \mathrm{A}^2 + \mathrm{B}$$
$$g\big(f(x)\big) = B(\mathrm{Ax} + \mathrm{B}) + \mathrm{A} = \mathrm{ABx} + \mathrm{B}^2 + \mathrm{A}$$
$$f\big(g(x)\big) - g(f(x) = B - A\text{から、}$$
$$(ABx + \mathrm{A}^2 + \mathrm{B}\ ) - (\mathrm{ABx} + \mathrm{B}^2 + \mathrm{A}\ ) = B - A\ 。$$
$$A^2 - B^2 + B - A = B - A\ 。$$
$$A^2 - B^2 = (A + B)(A - B) = 0$$
$A - B \neq 0$ より、$A + B = 0$ である。

問題4 (IMO 1964年 第2問)

a, b, cを三角形の辺の長さとする。この時、次の式を証明せよ。

$$a^2(b+c-a)+b^2\,(c+a-b)+c^2(a+b-c)\leq 3abc$$

証明

両辺を整理すると、

$$a(a-b)(a-c)+b(b-a)(b-c)+c(c-a)(c-b)\geq 0$$

この式は、a, b, c について対象なので、

$a\geq b\geq c\geq 0$ と仮定していい。

$$(a-b)\big(a(a-c)-b(b-c)\big)+c(a-c)(b-c)\geq 0$$

Shurの不等式

問題5 (AIME 1984年 第1問)

$a1, a2, a3, \ldots$ を公差1の等差数列とする。
$a1 + a2 + a3 \ldots + a98 = 137$ の時、
$a2 + a4 + a6 + a8 + \cdots + a98 = 93$ となることを証明せよ。

証明

公差1だから、$a(2n-1) = a(2n) - 1$。

奇数項をこの式で置き換える。

$a1 + a2 + a3 \ldots + a98$

$= a(2) - 1 + a(2) + a(4) - 1 + \mathrm{a}(4) + \cdots + \mathrm{a}(98) - 1 + \mathrm{a}98$

$= 2(a2 + a4 + a6 + a8 + \cdots + a98) - 49 = 137$

よって、$a2 + a4 + a6 + a8 + \cdots + a98 = (137 + 49)/2 = 93$

問題6 (IMO 1990年 Longlist 第77問)

PROBLEM 6

Adapted from IMO Longlist 1990 Problem 77[4]

For a, b, c reals, prove that $(a^2 + ab + b^2)(b^2 + bc + c^2)(c^2 + ca + a^2) \geq (ab + bc + ca)^3$.

</> FORMAL | INFORMAL

After cancelling terms appearing on both sides, we are left to prove that:

$$3a^2b^2c^2 + \sum_{sym} a^3b^2c \leq \sum_{cyc} a^4bc + \sum_{cyc} (a^4b^2 + b^4c^2)$$

After multiplying both sides by 2, we can rearrange the above inequality to:

$$0 \leq \sum_{cyc} (a^2b + a^2c - b^2c)^2$$

which clearly holds, giving the claim.

Open-AI Theorem Proverが行う証明

Open-AI Theorem Proverは、Leanによる証明を返す

「我々は、Leanのための
ニューラル定理証明システムを構築した。」

Open-AI Theorem Proverは、言語Leanによる証明を返す。

“Theorem Proving in Lean 4”
https://leanprover.github.io/theorem_proving_in_lean4/

Open-AI Theorem Proverが、証明した問題

問題1 (AMC12 2000年第5問)

$|x-2| = p$とする。$x < 2$の時、$x - p = 2 - 2p$ を証明せよ。

```
theorem amc12_2000_p5 -- ← theorem name
    (x p : ℝ)                        -- ← the statement we want
    (h₀ : x < 2)                     -- to prove
    (h₁ : abs (x - 2) = p) :
    x - p = 2 - 2 * p :=
begin                                -- ← formal proof starts here
   -- This first tactic requires that the prover invent
   -- the term: `abs (x - 2) = -(x - 2)`.
  have h₂ : abs (x - 2) = -(x - 2), {
     apply abs_of_neg,
     linarith,
  },
  rw h₁ at h₂,
  -- At this stage the remaining goal to prove is:
  -- `x - p = 2 - 2 * p` knowing that `p = -(x - 2)`.
  linarith,
end
```

theorem amc12_2000_p5

(x p : ℝ)
(h₀ : x < 2)
(h₁ : abs (x - 2) = p) :

x - p = 2 - 2 * p :=

問題

条件

証明のゴール

```
begin                                   -- ← formal proof starts here
  -- This first tactic requires that the prover invent
  -- the term: `abs (x - 2) = -(x - 2)`.
  have h₂ : abs (x - 2) = -(x - 2), {
    apply abs_of_neg,
    linarith,
  },
  rw h₁ at h₂,
  -- At this stage the remaining goal to prove is:
  -- `x - p = 2 - 2 * p` knowing that `p = -(x - 2)`.
  linarith,
end
```

```
theorem amc12_2000_p5 -- ← theorem name
    (x p : ℝ)                    -- ← the statement we want
    (h₀ : x < 2)                 -- to prove
    (h₁ : abs (x - 2) = p) :
    x - p = 2 - 2 * p :=
begin                            -- ← formal proof starts here
  -- This first tactic requires that the prover invent
  -- the term: `abs (x - 2) = -(x - 2)`.
  have h₂ : abs (x - 2) = -(x - 2), {
    apply abs_of_neg,
    linarith,
  },
  rw h₁ at h₂,
  -- At this stage the remaining goal to prove is:
  -- `x - p = 2 - 2 * p` knowing that `p = -(x - 2)`.
  linarith,
end
```

新しい仮説h2

h2の証明

マイナスの絶対値等式の証明

問題の証明

```
theorem amc12_2000_p5 -- ← theorem name
    (x p : ℝ)                        -- ← the statement we want
    (h₀ : x < 2)                     -- to prove
    (h₁ : abs (x - 2) = p) :
    x - p = 2 - 2 * p :=
begin                                -- ← formal proof starts here
  -- This first tactic requires that the prover invent
  -- the term: `abs (x - 2) = -(x - 2)`.
  have h₂ : abs (x - 2) = -(x - 2), {
    apply abs_of_neg,
    linarith,
  },
  rw h₁ at h₂,
  -- At this stage the remaining goal to prove is:
  -- `x - p = 2 - 2 * p` knowing that `p = -(x - 2)`.
  linarith,
end
```

問題の証明

h2 を使って h1を書き替える p = -(x - 2)

等式の証明

```
theorem amc12_2000_p5 -- ← theorem name
    (x p : ℝ)                        -- ← the statement we want
    (h₀ : x < 2)                     -- to prove
    (h₁ : abs (x - 2) = p) :
    x - p = 2 - 2 * p :=
begin                                -- ← formal proof starts here
   -- This first tactic requires that the prover invent
   -- the term: `abs (x - 2) = -(x - 2)`.
  have h₂ : abs (x - 2) = -(x - 2), {
     apply abs_of_neg,
     linarith,
  },
  rw h₁ at h₂,
  -- At this stage the remaining goal to prove is:
  -- `x - p = 2 - 2 * p` knowing that `p = -(x - 2)`.
  linarith,
end
```

Tactics

問題2 (AMC12B 2020年第6問)

$n \geq 9$なる全ての整数について、
$((n+2)! - (n+1)!)/n!$
は、完全平方数であることを示せ。

```
theorem amc12b_2020_p6
  (n : ℕ)
  (h0 : 9 ≤ n) :
  ∃ x : ℕ, (x:ℝ)^2 =
     (nat.factorial (n + 2) - nat.factorial (n + 1))
     / nat.factorial n :=
begin
  -- The model directly proposes `n + 1` as solution.
  use n + 1,
  field_simp [nat.factorial_ne_zero, pow_succ'],
  ring_exp
end
```

問題3 (Math dataset より)

$f(x) = Ax + B,\ g(x) = Bx + A$とする。$A - B \neq 0$で、
$f\big(g(x)\big) - g(f(x) = B - A$ なら、$\mathrm{A} + \mathrm{B} = 0$ を証明せよ。

```
theorem mathd_train_algebra_217
  (a b : ℝ)
  (f g : ℝ → ℝ)
  (h₀ : ∀ x, f x = a * x + b)
  (h₁ : ∀ x, f x = b * x + a)
  (h₂ : a ≠ b)
  (h₃ : ∀ x, f (g x) - g (f x) = b - a) : a + b = 0 :=
begin
  revert h₀ h₁ h₂ h₃,
  -- Initial contraposition.
  contrapose!,
  rintro ⟨h₀, ⟨h₁, h₂⟩⟩,
  -- The model proposes `0` as witness for the current
  -- goal that consists in `∃ (x : ℝ), f x ≠ a * x + b`.
  use (0 : ℝ),
  simp only [sub_eq_iff_eq_add, h₀, mul_zero, zero_add],
  norm_num at h₀,
end
```

問題4 (IMO 1964年 第2問)

a, b, cを三角形の辺の長さとする。この時、次の式を証明せよ。

$$a^2(b+c-a)+b^2\,(c+a-b)+c^2(a+b-c) \leq 3abc$$

```
theorem imo_1964_p2
  (a b c : ℝ)
  (h₀ : 0 < a ∧ 0 < b ∧ 0 < c)
  (h₁ : c < a + b)
  (h₂ : b < a + c)
  (h₃ : a < b + c) :
   a^2 * (b + c - a) + b^2 * (c + a - b) + c^2 * (a + b - c)
     ≤ 3 * a * b * c :=
begin
  -- Arguments to `nlinarith` are fully invented by our model.
  nlinarith [ sq_nonneg (b - a),
              sq_nonneg (c - b),
              sq_nonneg (c - a)]
end
```

問題5 (AIME 1984年 第1問)

$a1, a2, a3, \ldots$ を公差1の等差数列とする。
$a1 + a2 + a3 \ldots + a98 = 137$ の時、
$a2 + a4 + a6 + a8 + \cdots + a98 = 93$ となることを証明せよ。

```
theorem aime_1984_p1
  (u : ℕ → ℚ)
  (h₀ : ∀ n, u (n + 1) = u n + 1)
  (h₁ : ∑ k in finset.range 98, u k.succ = 137) :
  ∑ k in finset.range 49, u (2 * k.succ) = 93 :=
begin
  rw finset.sum_eq_multiset_sum,
  dsimp [finset.range] at h₁,
  simp [h₀], ring, norm_num at h₁,
  norm_num,
  apply eq_of_sub_eq_zero,
 { simp only [*, abs_of_pos, add_zero] at *,
  linarith },
end
```

問題6 (IMO 1990年 Longlist 第77問)

PROBLEM 6

Adapted from IMO Longlist 1990 Problem 77[4]

For a, b, c reals, prove that $(a^2 + ab + b^2)(b^2 + bc + c^2)(c^2 + ca + a^2) \geq (ab + bc + ca)^3$.

```
theorem imo_longlist_1990_p77
  (a b c : ℝ) :
  (a * b + b * c + c * a)^3 ≤
    (a^2 + a * b + b^2) *
    (b^2 + b * c + c^2) *
    (c^2 + c * a + a^2) :=
begin
  -- The three initial steps use Cauchy–Schwarz to prove
  -- `(a * b + b * c) ^ 2 ≤ (a ^ 2 + b ^ 2) * (b ^ 2 + c ^ 2)`
  -- which is required for the final call to `nlinarith`.
  let u : euclidean_space ℝ (fin 2) := ![a, b],
  let v : euclidean_space ℝ (fin 2) := ![b, c],
  have h₀ := real_inner_mul_inner_self_le u v,
  simp [u, v, fin.sum_univ_succ,
        ←pow_two, ←pow_two, le_of_lt, mul_assoc] at h₀,
```

```
  -- The model introduces another required cut (i.e. invent
  -- the term `0 ≤ (c + a) * (c + a)` and proves it).
  have h₃ : 0 ≤ (c + a) * (c + a),
  { nlinarith, },
  have h₄ := sq_nonneg (a * b + b * c + c * a),
  simp [sq, h₀, h₃, mul_add, add_mul] at h₄ ⊢,
  nlinarith [sq_nonneg (b - a),
             sq_nonneg (c - b),
             sq_nonneg (a - c)]
end
```

数学的証明の重要性とその難しさ

Deep Learningの到達点

「ディープラーニングは、言語・ビジョン・画像生成を含む多くの分野で目覚ましい成功を謳歌してきた。ディープラーニングがまだそれらの分野と比肩しうる成功を収めていない一つの領域がある。それは、広い見通しとシンボルに基いた推論を必要とするタスクの中にある。」

以下の引用は、すべて次の論文から。
Ilya Sutskever et al.
“Formal Mathematics Statement Curriculum Learning”
https://arxiv.org/pdf/2202.01344.pdf

二人対戦ゲーム

「二人対戦ゲームは、その例外である。ディープラーニング・システムは、このようなゲームでは、特に、セルフ・プレイでトレーニングされた場合、モンテカルロ木探索などの検索手順と組み合わせると、かなりの程度の推論能力を示す。しかし、ゲームが対象とするスコープは比較的狭いため、結果として達成される推論能力は制限されている。」

形式的推論能力の重要性

「この点では、インタラクティブな証明アシスタント、すなわち、形式的な数学での定理証明は、取り組むべき興味深いゲームのような領域に見える。その応用範囲が拡大しているためである。形式的な数学には、ゲームと同じように、軌跡（証明）が成功したかどうか（形式的に正しいか）を、自動的に判断する方法がある。しかし、形式的な数学の広い応用範囲を考えれば、（たとえば、数学的な推論の証明を見つける）そこで得られた強力な推論結果は、ゲームでの同等の結果よりも大きな意味がある。それは、（たとえば、ソフトウェア検証といった）重要な実際の問題にも適用できることを意味する。」

無限の動作空間

「形式的な数学には、非常に大きな検索スペース（たとえば、碁の場合のように）が存在するだけでなく、無限の動作空間がある。 証明の検索の各ステップで、モデルは、正常に動作する有限の動作セットからではなく、複雑で無限に存在するTactics（戦術）から動作を選択しなければならない。そこでは、システムの内部には存在しなかった数学的術語を、新たに生成して利用する必要がある可能性がある。（たとえば、証明の途中で利用される、「…を満たすxが存在する」のような、仮説witness を生成するとか、あるいは、証明の途中で導入されたこうした補題を論理的につなげていく cut ruleの利用とか。」

セルフ･プレイによる設定は存在しない

「形式的な数学では、証明者は対戦相手と対戦するのではなく、証明するための一連の数学的命題と対戦する。難しすぎる命題に直面した場合、証明者が最初に取り組むための中間のより簡単な命題を生成できるようにする、環境の再構成の簡単な手段は、形式的数学には存在しない。数学敵証明とゲームとのこの非対称性は、2人ゲームで一般的に使用される対称的なセルフ･プレイ･アルゴリズムを、数学的証明に単純に適用することを妨げる。

難しさにどう立ち向かうか？

二つの難しさにどう立ち向かうか？

二つの難しさ

1. 「無限の動作空間」
2. 「セルフ･プレイによる環境づくりは存在しない」

強化学習の方法を形式的な数学に
素朴に適用することは成功しそうにない。
この論文では、二つ目の問題にフォーカスする

以下の引用は、すべて次の論文から。
Ilya Sutskever et al.
“Formal Mathematics Statement Curriculum Learning”
https://arxiv.org/pdf/2202.01344.pdf

セルフ･プレイが果たした役割と
それに代わるもの

「二つ目の問題に取り組む上で基礎となるのは、セルフ･プレイのもっとも重要な役割は、それが教師なし学習のカリキュラムを提供していたという観察である。我々は、それらの代わりに、様々のレベルの難しさからなる補助的な問題群を（証明を要求することなく）提供することを提案する。」

我々が、経験的に示すことが出来たこと

「これらの補助的な問題の難易度が、十分に多様なものであれば、単純な学習の繰り返しの手順で、だんだんと難しくなる問題のカリキュラムを解くことができ、最終的には、目標の分布に一般化できることを、我々は経験的に示した。このアプローチは、自動的に生成された、あるいは、人間によって編集された補助的な問題の分布の双方で機能することを、我々は示した。こうして、これを活用して、miniF2Fベンチマークで最先端の成果を実現した。」

我々の結果が示唆すること

「我々の結果は、形式的数学が絶えず継続的にその姿を改善していることは、今回は部分的に手動で行った形式的命題の集合を生成する問題に、潜在的に還元できることを示唆している。ただ、将来的には、この作業は、最終的にはより多くの自動化で拡張できる可能性がある（特定の分野にもっと特化した命題生成器とか、自然言語から形式的命題への機械翻訳という形で）。

我々のモデルの限界

Appendix F.1 で議論したように、cutやwitness を生成する能力を我々のモデルは持つにもかかわらず、現在のその主要な限界は、（我々が提案した検索手順のもとでは）二つあるいは三つ以上の自明ではない数学的推論をつなげていくことの無能力さにあると信じている。このことが、（例外的にではなく）整合的に数学オリンピックの問題に挑戦することを妨げている。

Despite our models' capability, as discussed in Appendix F.1, to generate cuts and witnesses, we believe that their current main limitation lies in their inability (under our proposed search procedure) to chain more than 2 or 3 non-trivial steps of mathematical reasoning, preventing them from consistently (instead of exceptionally) solving challenging olympiad problems..

我々のモデルの限界

我々は、何度も我々のモデルが生成した証明手順の複雑さに、強く印象付けられてきた。しかし、こうした推論ステップを必要とする証明の多くは、現在のコンピュータの能力の地平を超えたところにある。我々が、たとえ挑戦的な数学オリンピックの選択された問題を解いたとしても、我々のモデルは、いまだこれらの競技の最も優秀な学生たちと競争するにははるかに遠い場所にいるのである。

We've been repeatedly impressed by the complexity of some of the proofsteps generated by our models. But, proofs requiring many of such reasoning steps remain beyond our current compute horizon. Even if we solved a selection of challenging olympiad problems, our models are still very far from being competitive with the brightest students in these competitions

我々のモデルの限界

我々のモデルはcutを生成するいくらかの能力を示したものの、彼らが生成したcutはたいていの場合、浅いものだった。（それらは、ごく少数の証明ステップ（tacticのこと）しか含まず、必ずしも、証明の構造を深く変えることはなかった。）

While our models have demonstrated some capabilities to generate cuts, the cuts they generate are often shallow (they involve only a few proofsteps and don’t necessarily deeply change the structure of the proof


50

第二部

Open-AI Theorem Proverは何を学習したのか？

Open-AI Theorem Proverは
何を学習したのか

数学的証明の場合には、ゲームの場合のように、セルフ・プレイで訓練用データをいくらでも提供できるわけではないことを先に見て来た。

それでは、Open-AI Theorem Proverには、まず最初に、どのような訓練用データが提供されたのだろうか？ ここでは、それをみていこう。

もちろん、このデータを選択し提供したのは、人間である。

三つの学習カリキュラム

Open-AI Theorem Prover に与えられたのは、次の三つの訓練用データである。

1. miniF2F-curriculum
2. synth-ineq
3. mathlib-train

以下、それがどんなものかを見ていこう。
（今回は、miniF2F-curriculum を紹介する）

第二部
Open-AI Theorem Proverは何を学習したのか？

彼は、何を学習したのか？ (1)
miniF2F-curriculum

彼は、何を学習したのか？ (2)
synth-ineq

彼は、何を学習したのか？ (3)
mathlib-train

彼は、何を学習したのか？（1）
miniF2F-curriculum

miniF2F-curriculum

miniF2F-curriculum の主要部分は、数学オリンピックにチャレンジしようとする中高生向けに、LehoczkyとRusczykがつくった問題集 “The Art of Problem Solving” からの抜粋である。327の問題が選ばれている。もちろん、それはLean語に変換されている。

https://github.com/openai/miniF2F/blob/statement_curriculum_learning/lean/src/statement_curriculum_learning/aopsbooks.lean に、公開されている。

```
40  theorem aopsbook_v1_c29_p567
41    (x y : ℝ)
42    (h₀ : x ≠ 0 ∧ y ≠ 0)
43    (h₁ : 2 / x = y / 3)
44    (h₂ : y / 3 = x / y) :
45    x^3 = 12 :=
46  begin
47    sorry
48  end
49
50  theorem aopsbook_v2_c14_em2
51    (a b : ℝ)
52    (h₀ : 0 < a ∧ 0 < b) :
53    real.sqrt (a * b) ≤ (a + b) / 2 :=
54  begin
55    sorry
56  end
57
58  theorem aopsbook_v2_c8_p132_1
59    (f : ℝ → ℝ)
60    (h₀ : ∀ x ≠ 0, f x = (real.sin (3 * x)) / (6 * x)) :
61    filter.tendsto f filter.at_top (𝓝 0) :=
62  begin
63    sorry
64  end
```

miniF2F-curriculumの一部

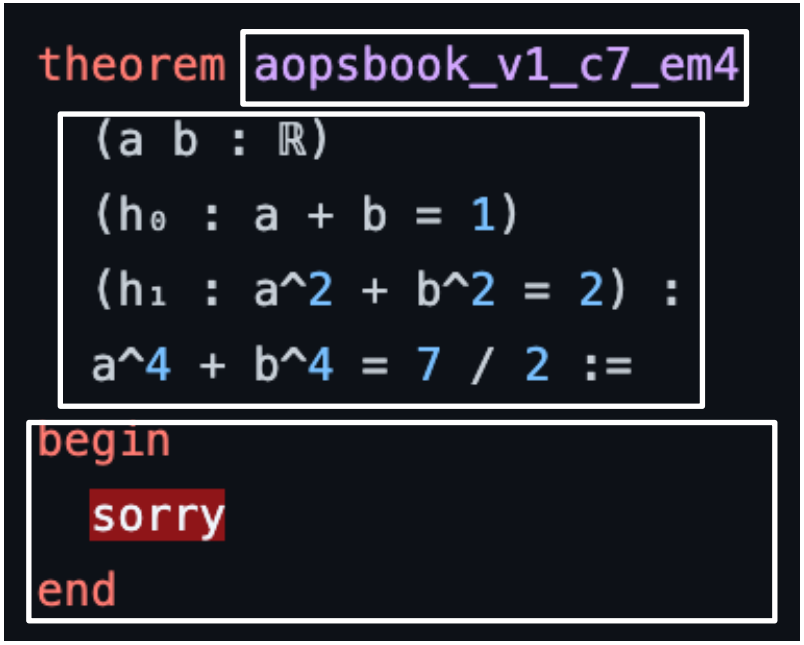
```
2783
2784  theorem aopsbook_v2_c6_ex4
2785    (P : polynomial ℚ)
2786    (h₀ : aeval (3-complex.I) P = 0)
2787    (h₁ : aeval (4+real.sqrt 2) P = 0) :
2788    aeval (3+complex.I) P = 0 ∧ aeval (2-real.sqrt 2) P = 0 :=
2789  begin
2790    sorry
2791  end
2792
2793  theorem aopsbook_v2_c6_p101
2794    (P Q : polynomial ℝ)
2795    (h₀ : P = (X^2 - (C 3)*X - C 2)^2 - (C 3)*(X^2 - (C 3)*X - C 2) - (C 2) - X)
2796    (h₁ : Q = X^2 - (C 4)*X - C 2) :
2797    ∀ a, is_root Q a → is_root P a :=
2798  begin
2799    sorry
2800  end
2801
2802  theorem aopsbook_v2_c6_p97
2803    (q₁ r₁ q₂ r₂ : polynomial ℝ)
2804    (h₀ : X^8 / (X + (C 1/2)) = q₁)
2805    (h₁ : X^8 / (X + (C 1/2)) = r₁)
2806    (h₂ : q₁ / (X + (C 1/2)) = q₂)
2807    (h₃ : q₁ % (X + (C 1/2)) = r₂) :
2808    r₂ = C (-1/16) :=
2809  begin
2810    sorry
2811  end
```

miniF2F-curriculumの一部

彼は、何を学習したのか？

```
theorem aopsbook_v1_c7_em4
  (a b : ℝ)
  (h₀ : a + b = 1)
  (h₁ : a^2 + b^2 = 2) :
  a^4 + b^4 = 7 / 2 :=
begin
  sorry
end
```

彼は、何を学習したのか？



定理の名前

“aops” は本の名前

“The Art of Problem Solving”

定理の内容

定理の証明

“sorry” は、証明が省略されていることを表す

彼は、何を学習したのか？

定理の前提部分

```
theorem aopsbook_v1_c7_em4
  (a b : ℝ)
  (h₀ : a + b = 1)
  (h₁ : a^2 + b^2 = 2) :
  a^4 + b^4 = 7 / 2 :=
begin
  sorry
end
```

定理の結論部分

どのような定理か

a, bを実数とする
a + b = 1で,
$a^2 + b^2 = 2$ なら
$a^4 + b^4 = 7/2$ である

彼は、何を学習したのか？（2）
synth-ineq

synth-ineq

Appendix D. Synthetic inequalities

"Formal Mathematics Statement Curriculum Learning"

https://arxiv.org/pdf/2202.01344.pdf

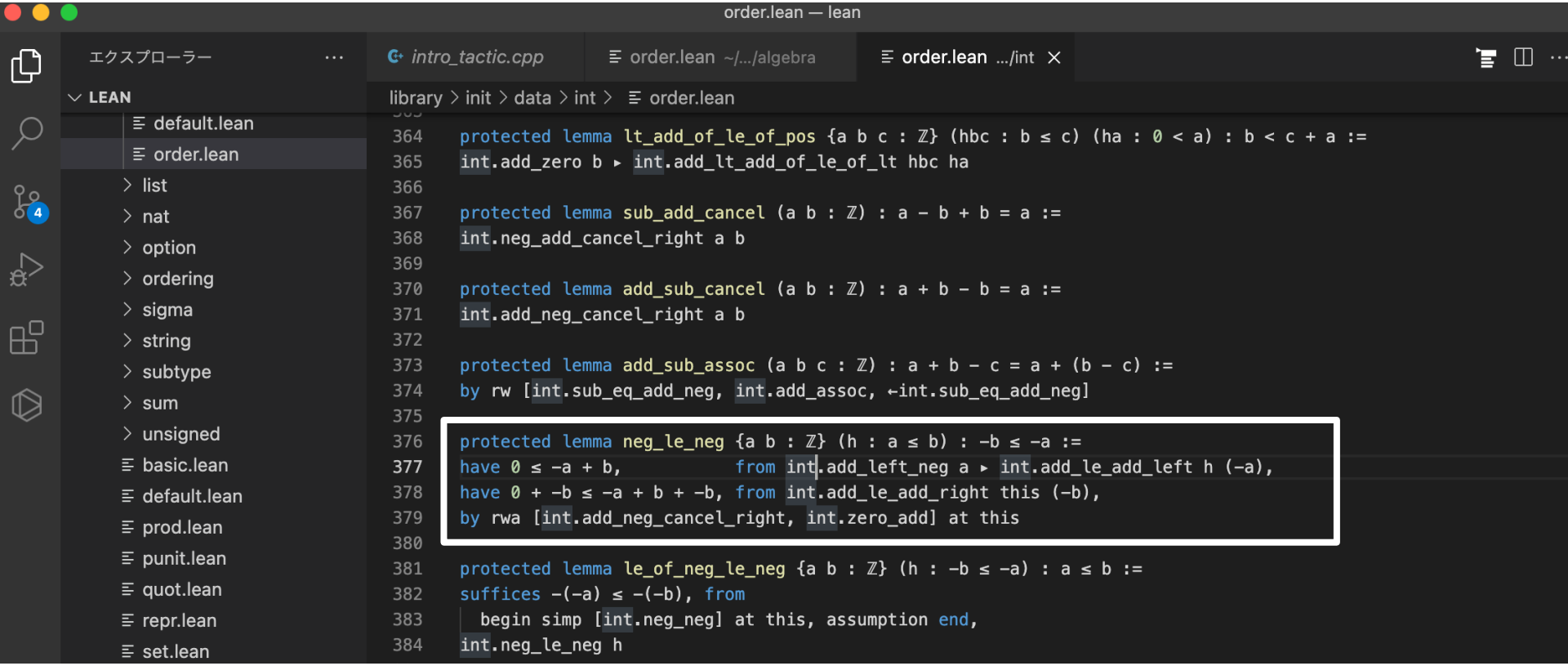
- Seed expressions generation
- Inequality composition
- Simplification

List of inequality composition theorems

- neg_le_neg
- inv_le_inv
- mul_self_le_mul_self
- div_le_one_of_le

- mul_le_mul
- add_le_add
- div_le_div
- mul_le_mul_of_nonneg
- le_mul_of_ratio

lemma neg_le_neg
{a b : ℤ} (h : a ≤ b) : -b ≤ -a



lean/library/init/data/int/order.lean

- lemma **neg_le_neg**
  {a b : ℤ} (h : a ≤ b) : -b ≤ -a
  lean/library/init/data/int/order.lean 他。

- lemma **inv_le_inv**
  (ha : 0 < a) (hb : 0 < b) : $a^{-1} \le b^{-1} \leftrightarrow b \le a$
  mathlib/src/algebra/order/field.lean 他。

- theorem **mul_self_le_mul_self**
  {n m : ℕ} (h : n ≤ m) : n * n ≤ m * m
  mathlib/src/data/nat/basic.lean 他。

- lemma **div_le_one_of_le**
  (h : a ≤ b) (hb : 0 ≤ b) : a / b ≤ 1
  mathlib/src/algebra/order/field.lean 他。

Examples 1

- AmGm a b
  (67:R) ((1:R)/(10:R)) ((1:R)/(10:R))
  ((8:R)/(10:R))
- theorem
  synthetic_ineq_nb_seed_var_0_depth_0_p_1
  (a b : R) (h0 : 0 < a) (h1 : 0 < b) :
  (67:R) ^ ((8:R) / (10:R)) * b ^ (10:R) ⁻¹ * a ^
  (10:R) ⁻¹ ≤ (8:R) / (10:R) * (67:R) + (10:R) ⁻¹ *
  a + b * (10:R) ⁻¹

$$67^{\frac{8}{10}} b^{\frac{1}{10}} a^{\frac{1}{10}} \leq \frac{8}{10} 67 + \frac{1}{10} a + \frac{1}{10} b$$

$$x^l y^m z^n \leq lx + my + nz$$

Examples 2

- Sqnonneg a ((a) + ((-68:R)))
- theorem
  synthetic_ineq_nb_seed_var_4_depth_0_p_4
  (a b : R)
  (h0 : 0 < a)
  (h1 : 0 < b) :
  (2:R) * (a * (a + -(68:R))) ≤ (a + -(68:R)) ^ 2 + a ^ 2

$$2a(a-68) \leq (a-68)^2 + a^2$$

Examples 3

- AddLeAdd
  Bernoulli 99 c
  AddLeAdd
  SelfDivConst ((a) / (f)) 6
  LeMulOfRatio
  SelfDivConst c 70
  DivLeDiv
  Cauchy ((a) / (f)) d c (log (((59:R) + f)))
  Young ((a) / (f)) a ((3:R)/(2:R)) ((3:R)/(1:R))

- theorem
  synthetic_ineq_nb_seed_var_4_depth_4_p_13
  (a b c d e f : R)
  (h0 : 0 < a) (h1 : 0 < b) (h2 : 0 < c)
  (h3 : 0 < d) (h4 : 0 < e) (h5 : 0 < f) :
  (1:R) + (99:R) * c + (a / f / (6:R) + a * (a / f) / ((d ^ 2 + a ^ 2 / f ^ 2) * (real.log ((59:R) + f) ^ 2 + c ^ 2))) ≤ ((a / f) ^ ((3:R) / (2:R)) / ((3:R) / (2:R)) + a ^ 3 / (3:R)) / (real.log ((59:R) + f) * d + a / f * c) ^ 2 * (c / (c / (70:R))) + a / f + (c + (1:R)) ^ 99

INT

INT: AN INEQUALITY BENCHMARK FOR EVALUATING GENERALIZATION IN THEOREM PROVING

https://arxiv.org/pdf/2007.02924.pdf

https://github.com/albertqjiang/INT

Abstract

In learning-assisted theorem proving, one of the most critical challenges is to generalize to theorems unlike those seen at training time.

In this paper, we introduce INT, an INequality Theorem proving benchmark designed to test agents' generalization ability. INT is based on a theorem generator, which provides theoretically infinite data and allows us to measure 6 different types of generalization, each reflecting a distinct challenge, characteristic of automated theorem proving.

BENCHMARKING SIX DIMENSIONS OF GENERALIZATION

- IID Generalization
- Initial Condition
- Axiom Orders
- Axiom Combinations
- Number of Axioms
- Proof Length
- Generalization

visualize_rl.ipynb — INT

エクスプローラー (⇧⌘E)

INT

- algos
- data_generation
- ineqVis
- legacy
- logic
- proof_system
- representation
- visualization
  - __init__.py
  - latex_parse.py
  - model_debug.py
  - proof_by_obj.py
  - seq_parse.py
  - supervised_learning_comp...
  - visualize_rl.ipynb
- .gitignore
- README.md
- test_representation.py

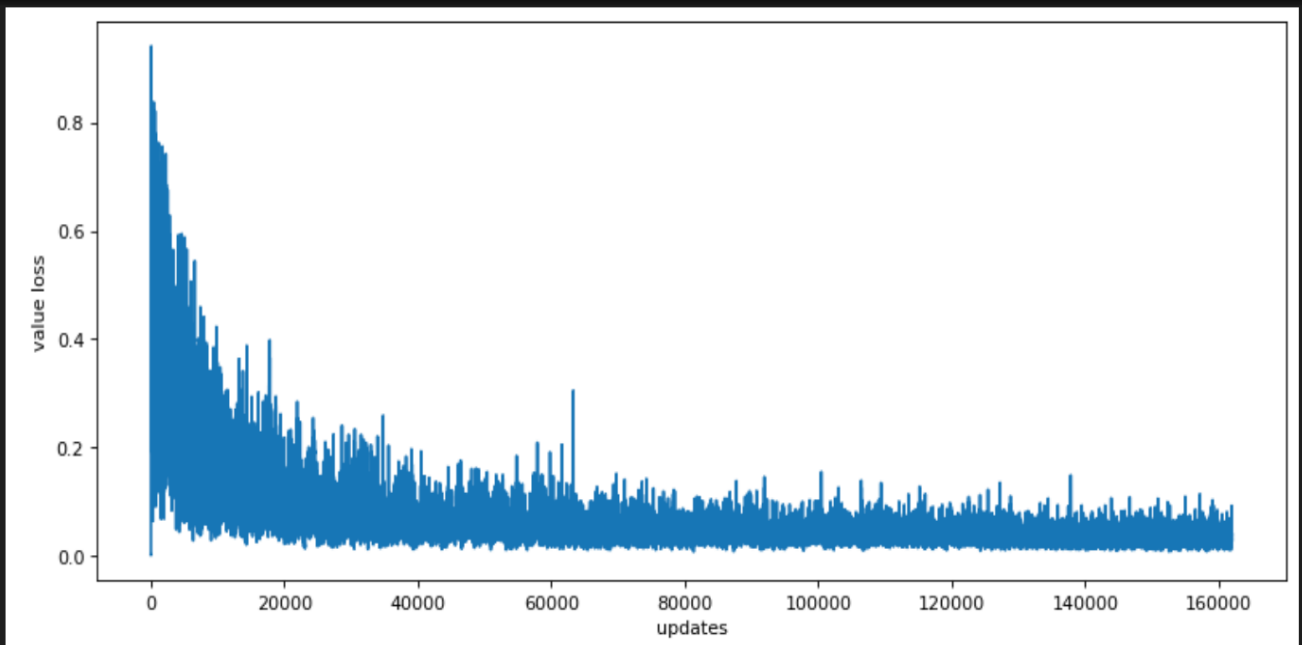
アウトライン

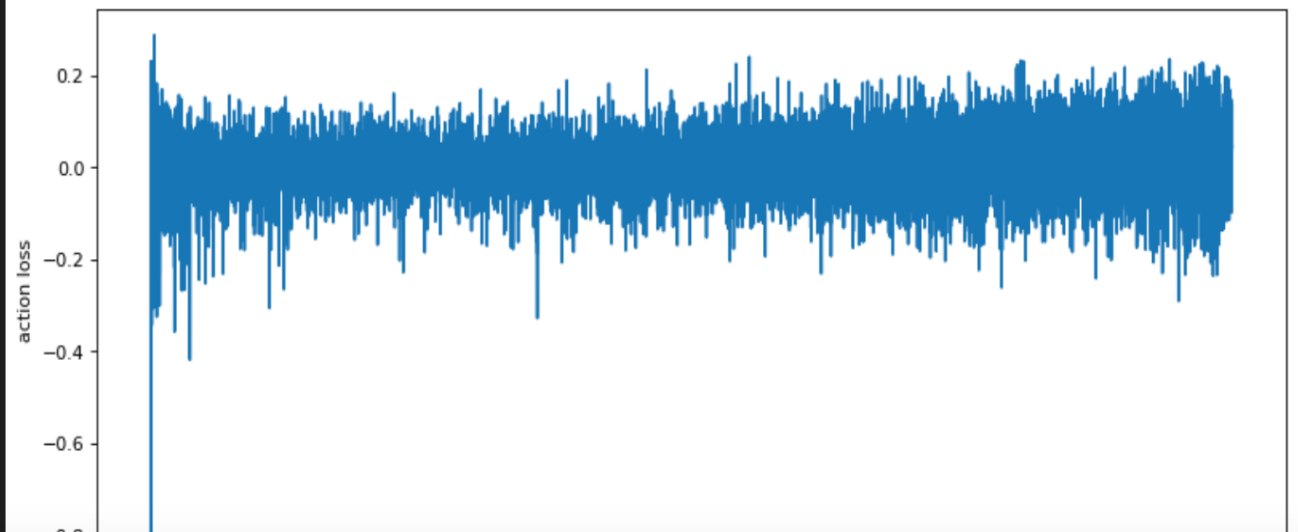
タイムライン

visualize_rl.ipynb

visualization > visualize_rl.ipynb > from baselines.common import plot_util as pu

＋ コード ＋ マークダウン | すべてを実行 すべてのセルの出力をクリアする | ... カーネルの選択





彼は、何を学習したのか？（3）
mathlib-train

mathlib-train

mathlib-train は、量的にも質的にも、Open-AI Theorem Proverが学習する訓練用データの本体だと思う。

量的には、

- miniF2F-curriculum　327
- synth-ineq　5,600
- mathlib-train　25,000 (25K)

質的にも、mathlib-train は一味違っている。

それは、多くの数学の領域をカバーしている、数学の定理集（証明付き）である。

mathlibがカバーしている領域

~/miniF2F/_target/deps/mathlib/src

algebra
algebraic_geometry
algebraic_topology
analysis
category_theory
combinatorics
computability
control
data
dynamics
field_theory
geometry
group_theory
linear_algebra
logic
measure_theory
meta
model_theory
number_theory
order
probability_theory
representation_theory
ring_theory
set_theory
tactic
testing
topology

~/miniF2F/_target/deps/mathlib/src/algebra

```
abs.lean
abs.olean
add_torsor.lean
add_torsor.olean
algebra
associated.lean
associated.olean
big_operators
bounds.lean
bounds.olean
category
char_p
char_zero.lean
char_zero.olean
continued_fractions
covariant_and_contravariant.lean
covariant_and_contravariant.olean
indicator_function.lean
indicator_function.olean
invertible.lean
invertible.olean
is_prime_pow.lean
iterate_hom.lean
iterate_hom.olean
lie
linear_recurrence.lean
linear_recurrence.olean
module
monoid_algebra
ne_zero.lean
ne_zero.olean
non_unital_alg_hom.lean
non_unital_alg_hom.olean
opposites.lean
```

~/miniF2F/_target/deps/mathlib/src/algebra

```
cubic_discriminant.lean                    opposites.olean
default.lean                               order
direct_limit.lean                          pempty_instances.lean
direct_sum                                 periodic.lean
divisibility.lean                          periodic.olean
divisibility.olean                         pointwise.lean
dual_number.lean                           pointwise.olean
euclidean_domain.lean                      polynomial
euclidean_domain.olean                     punit_instances.lean
field                                      punit_instances.olean
field_power.lean                           quadratic_discriminant.lean
field_power.olean                          quadratic_discriminant.olean
free.lean                                  quandle.lean
free_algebra.lean                          quaternion.lean
free_monoid.lean                           quaternion_basis.lean
free_monoid.olean                          quotient.lean
free_non_unital_non_assoc_algebra.lean     quotient.olean
gcd_monoid                                 regular
geom_sum.lean                              ring
```

~/miniF2F/_target/deps/mathlib/src/algebra

```
geom_sum.lean
geom_sum.olean
graded_monoid.lean
group
group_action_hom.lean
group_action_hom.olean
group_power
group_ring_action.lean
group_ring_action.olean
group_with_zero
hierarchy_design.lean
homology
ring
ring_quot.lean
smul_with_zero.lean
smul_with_zero.olean
squarefree.lean
squarefree.olean
star
support.lean
support.olean
triv_sq_zero_ext.lean
tropical
```

```
/-
Copyright (c) 2018 Kenny Lau. All rights reserved.
Released under Apache 2.0 license as described in the file LICENSE.
Authors: Kenny Lau, Yury Kudryashov
-/
import algebra.module.basic
import linear_algebra.basic
import tactic.abel
import data.equiv.ring_aut

/-!
# Algebras over commutative semirings

In this file we define associative unital `algebra`s over commutative (semi)rings, algebra
homomorphisms `alg_hom`, and algebra equivalences `alg_equiv`.

`subalgebra`s are defined in `algebra.algebra.subalgebra`.

For the category of `R`-algebras, denoted `Algebra R`, see the file
`algebra/category/Algebra/basic.lean`.

See the implementation notes for remarks about non-associative and non-unital algebras.

## Main definitions:

* `algebra R A`: the algebra typeclass.
* `alg_hom R A B`: the type of `R`-algebra morphisms from `A` to `B`.
* `alg_equiv R A B`: the type of `R`-algebra isomorphisms between `A` to `B`.
* `algebra_map R A : R →+* A`: the canonical map from `R` to `A`, as a `ring_hom`. This is the
  preferred spelling of this map.
```

Appendix E.
Example proofs from mathlib-train

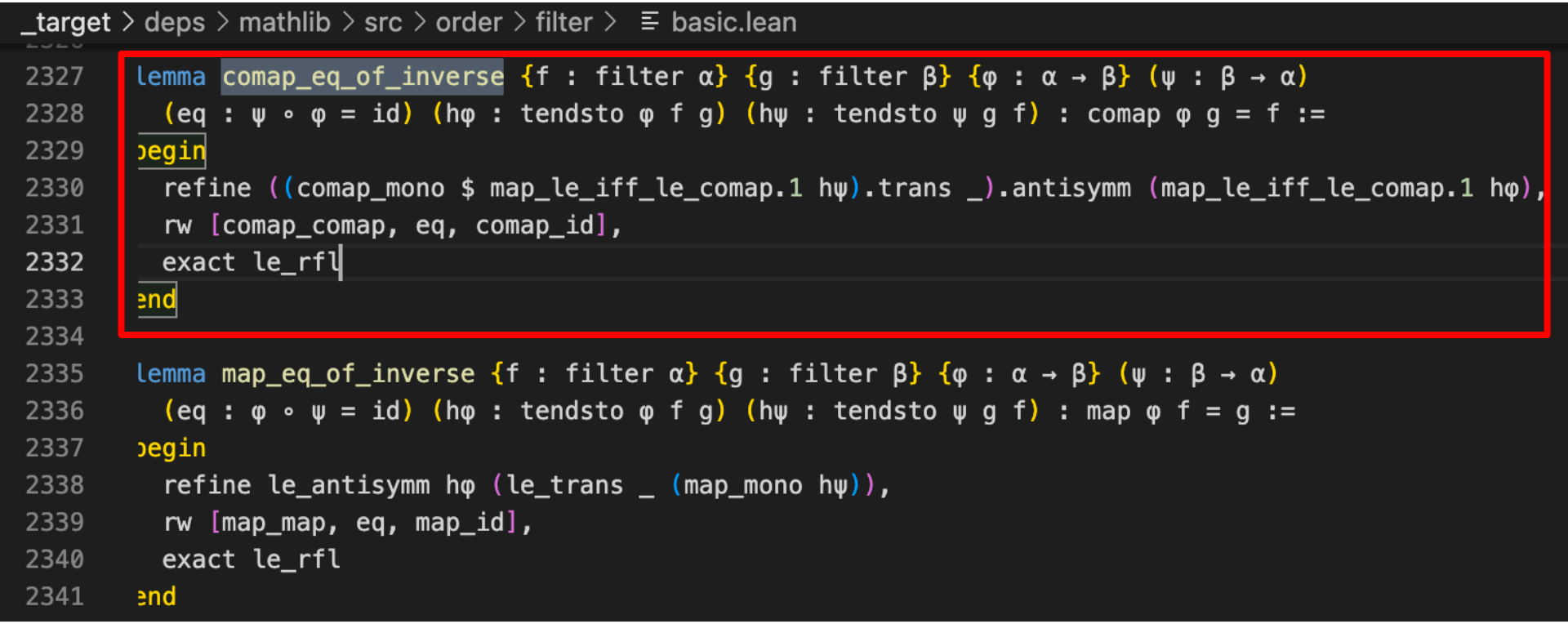
lemma comap_eq_of_inverse

```
lemma comap_eq_of_inverse {f : filter α} {g : filter β}
  {ϕ : α → β} (ψ : β → α) (eq : ψ ∘ ϕ = id)
  (hϕ : tendsto ϕ f g) (hψ : tendsto ψ g f) :
  comap ϕ g = f :=
```

```
begin
  refine ((comap_mono $
    map_le_iff_le_comap.1 hψ).trans _).antisymm
    (map_le_iff_le_comap.1 hϕ),
  rw [comap_comap, eq, comap_id],
  exact le_rfl
end
```

lemma comap_eq_of_inverse

mathlib/src/order/filter/basic.lean



lemma comap_eq_of_inverse

```
lemma comap_eq_of_inverse {f : filter α} {g : filter β}
  {ϕ : α → β} (ψ : β → α) (eq : ψ ∘ ϕ = id)
  (hϕ : tendsto ϕ f g) (hψ : tendsto ψ g f) :
  comap ϕ g = f :=
```

```
begin
  refine ((comap_mono $
    map_le_iff_le_comap.1 hψ).trans _).antisymm
    (map_le_iff_le_comap.1 hϕ),
  rw [comap_comap, eq, comap_id],
  exact le_rfl
end
```

```
begin
  refine le_antisymm _ (filter.map_le_iff_le_comap.1 hϕ),
  refine λ s hs, _,
  rw mem_comap,
  use [ψ ⁻¹' s, hψ hs],
  rw [← preimage_comp, eq, preimage_id]
end
```

lemma sum_range_sub_sum_range

```
lemma sum_range_sub_sum_range {α : Type*} [add_comm_group α]
  {f : ℕ → α} {n m : ℕ} (hnm : n ≤ m) :
  ∑ k in range m, f k - ∑ k in range n, f k =
  ∑ k in (range m).filter (λ k, n ≤ k), f k :=

begin
  rw [← sum_sdiff (@filter_subset _ (λ k, n ≤ k) _
    (range m)), sub_eq_iff_eq_add,
    ← eq_sub_iff_add_eq, add_sub_cancel'],
  refine finset.sum_congr
    (finset.ext $ λ a, ⟨λ h, by simp at *; finish,
    λ h, have ham : a < m :=
      lt_of_lt_of_le (mem_range.1 h) hnm,
      by simp * at *⟩)
    (λ _ _, rfl)
end
```

```
begin
  rw [← sum_Ico_eq_sub _ hnm],
  congr,
  apply finset.ext,
  simp [Ico.mem, *],
  tauto
end
```

lemma prod_inv_distrib

```
lemma prod_inv_distrib : (∏ x in s, (f x)⁻¹) =
  (∏ x in s, f x)⁻¹ :=
```

```
begin
  classical,
  by_cases h : ∃ x ∈ s, f x = 0,
  { simpa [prod_eq_zero_iff.mpr h, prod_eq_zero_iff]
      using h },
  { push_neg at h,
    have h' := prod_ne_zero_iff.mpr h,
    have hf : ∀ x ∈ s, (f x)⁻¹ * f x = 1 := λ x hx,
    inv_mul_cancel (h x hx),
    apply mul_right_cancel' h',
    simp [h, h', ← finset.prod_mul_distrib,
      prod_congr rfl hf] }
end
```

```
begin
  classical; induction s using
    finset.induction_on with a s has ih,
  { simp, },
  simp only [has, prod_insert has, mul_inv_rev'],
  finish
end
```


50

第三部

数学者、証明にコンピュータを使い始める

数学者、コンピュータを使い始める

これまでの第一部・第二部では、コンピュータに証明問題を解かせる試みとして、Open-AI Theorem Proverを紹介してきた。

この第三部では、数学者が証明問題を解くのに、コンピュータを利用し始めていることを紹介して、数学へのコンピュータ利用の背景とその意味を考える。

前者では、問題を解く主体はコンピュータだが、後者では、問題を解く主体は、人間＝数学者である。

第三部
数学者、証明にコンピュータを使い始める

「連続体仮説」をコンピュータで解く

Feit–Thompson定理の形式的証明

ケプラー予想の証明

連続体仮説をコンピュータで解く

連続体仮説の形式的な独立性証明

2020年、Jesse Michael Han, Floris van Doornは、連続体仮説の集合論ZFからの独立性の形式的証明 – コンピュータによる証明を与えた。

“A Formal Proof of the Independence of the Continuum Hypothesis”
https://arxiv.org/pdf/2102.02901.pdf

その証明は、GitHubで公開されている。
https://github.com/flypitch/flypitch/

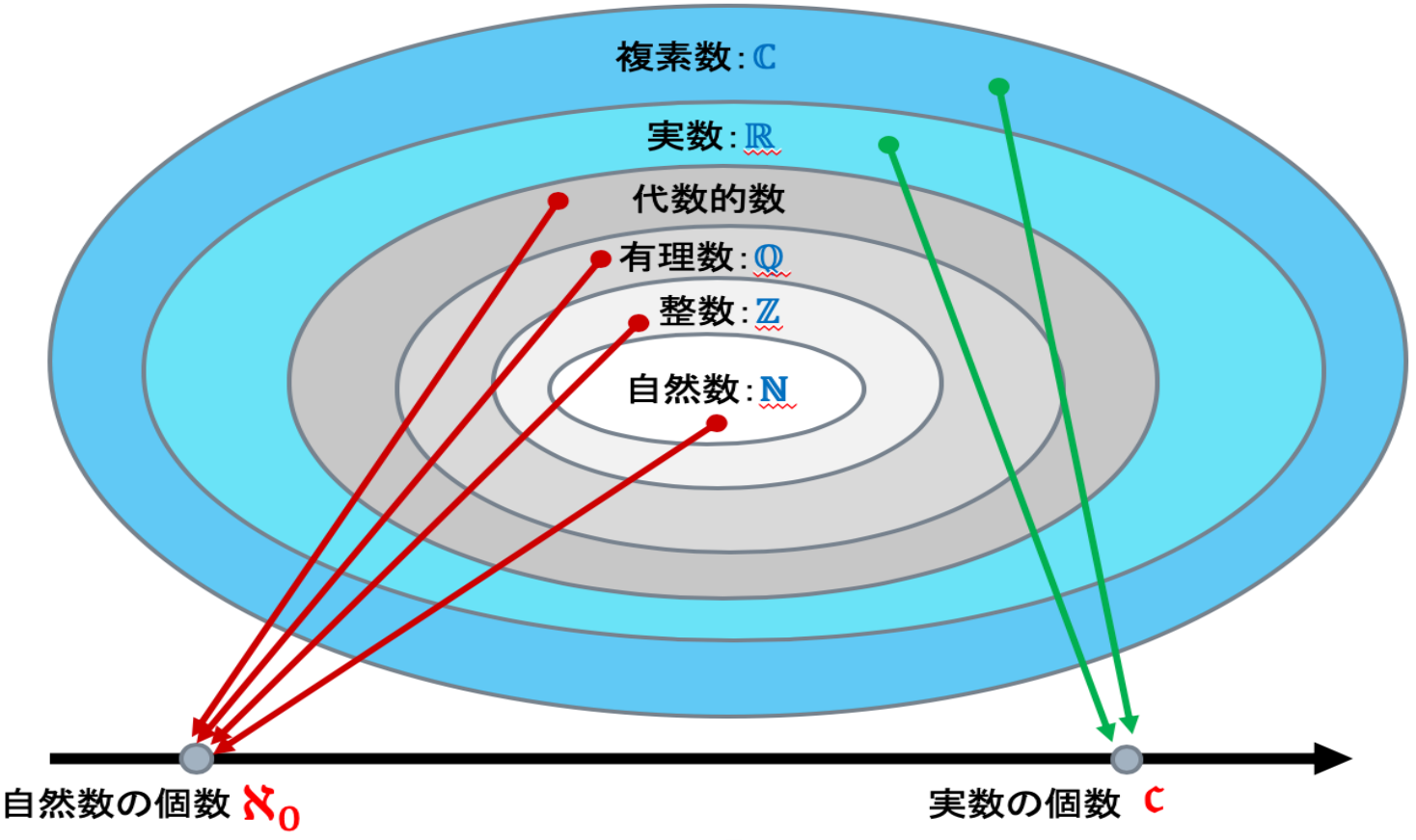
Jesse Michael Han

Researcher at @OpenAI

Math PhD at the University of Pittsburgh, interested in formal proofs and applying deep learning to automated theorem proving.

連続体仮説とは何か？

$\mathfrak{c} = 2^{\aleph_0}$ ？



カントール　連続体仮説 1878年

カントールは、彼が見つけた可算無限の濃度$\aleph_0$より大きな実数の無限の濃度 $\mathfrak{c}$を、無限濃度の階層の次の無限$\aleph_1$に等しいと見当をつけた。またこれが、$2^{\aleph_0}$に等しいと予想した。
これが、カントールの「連続体仮説 CH(Continuum Hyposesis)」である。

$$\mathfrak{c} = \aleph_1 = 2^{\aleph_0}$$

さらに、一般に、次の関係が成り立つとした。それを「一般連続体仮説 GCH(General Continuum Hyposesis) 」という

$$\aleph_{n+1} = 2^{\aleph_n}$$

しかし、彼はこの問題を解くことが出来なかった。

ヒルベルトの23の問題　1900年

ヒルベルトは、1900年に、20世紀の数学が解くべき23の問題のリストを発表した。その問題提起は、20世紀の数学に少なくない影響を与えた。カントールの「連続体仮説」は、この23の問題の第一番目の問題に取り上げられている。

ゲーデル　1940年
連続体仮説と集合論の無矛盾性の証明

「世紀の難問」と呼ばれた連続体仮説の問題で、大きな前進があったのは、1940年のことだ。クルト・ゲーデルが、連続体仮説と選択公理の二つを満たす集合論のモデルを実際に構成して見せたのだ。この集合論のモデルの中では、連続体仮説は「真」である。

この結果は、連続体仮説を直接に証明したものではないことに注意しよう。それは、集合論と連続体仮説には矛盾がないこと、すなわち、集合論の中では連続体仮説の否定は証明できないことを意味している。

コーエン　1963年
連続体仮説と集合論の独立性の証明

連続体仮説を最終的に解決したのは、J.P.コーエンだった。彼は、「強制法(Forcing Method)」というモデル構成法を創出し、連続体仮説の否定が成り立つモデルを構成した。

先のゲーデルの結果と合わせると、連続体仮説は、集合論の公理群からは独立であることが証明されたことになる。連続体仮説は、集合論の公理群からは証明できないのだ。

結果から見ると、カントールが、全力で連続体仮説を証明しようとして、成功しなかったのには理由があったのである。

非カントール的集合論の発見

コーエンの結果は、始祖のカントールが構想した、いわばカントール的集合論とは異なる、非カントール的集合論が存在することを明らかにした衝撃的なものだった。

ユークリッド幾何学の「第五公準」が、他のユークリッド幾何学の公理群からは独立で、それらから証明できないことの認識の成立が、非ユークリッド幾何学を成立させたのと同じことが、「数学の基礎」である集合論で起きたのである。

コーエンの結果とその方法は、1960年代の中頃には、集合論の構造についての認識を飛躍的に発展させた。

証明をコンピュータで実行する

```
git clone https://github.com/flypitch/flypitch.git
leanproject get-mathlib-cache
leanproject build
lean --trust=0 ./src/summary.lean
```

leanproject get-mathlib-cache
証明に必要なライブラリーの準備

maruyama@MacBook-2 flypitch % leanproject get-mathlib-cache

configuring flypitch 2.2

mathlib: cloning https://github.com/leanprover-community/mathlib to _target/deps/mathlib

> mkdir -p _target/deps/mathlib

> git clone https://github.com/leanprover-community/mathlib _target/deps/mathlib

Cloning into '_target/deps/mathlib'...

remote: Enumerating objects: 254227, done.

remote: Counting objects: 100% (35/35), done.

remote: Compressing objects: 100% (22/22), done.

remote: Total 254227 (delta 17), reused 27 (delta 13), pack-reused 254192

Receiving objects: 100% (254227/254227), 132.54 MiB | 1.21 MiB/s, done.

Resolving deltas: 100% (202695/202695), done.

```
Checking out files: 100% (2566/2566), done.
> git checkout --detach 883d974b608845bad30ae19e27e33c285200bf84    # in directory _target/deps/mathlib
HEAD is now at 883d974b6 feat(algebra/module): sum_smul' (for semimodules) (#1752)
Looking for mathlib oleans for 883d974b6
  locally...
  remotely...
  Found remote mathlib oleans
Located matching cache
  883d974b6: 100%|████████████████████████████████| 15.4M/15.4M [00:14<00:00, 1.09MB/s]
Applying cache
  files extracted: 1256 [00:04, 307.05/s]
maruyama@MacBook-2 flypitch %
```

leanproject build
証明プロジェクトのビルド

```
maruyama@MacBook-2 flypitch % leanproject build
Building project flypitch
configuring flypitch 2.2
mathlib: trying to update _target/deps/mathlib to revision
883d974b608845bad30ae19e27e33c285200bf84
> git checkout --detach 883d974b608845bad30ae19e27e33c285200bf84
# in directory _target/deps/mathlib
HEAD is now at 883d974b6 feat(algebra/module): sum_smul' (for
semimodules) (#1752)
> lean --make src
/Users/maruyama/flypitch/_target/deps/mathlib/src/tactic/restate_axiom.le
an:
par/Users/maruyama/flypitch/_target/deps/mathlib/src/algebra/group/free
_monoid.lean/Users/maruyama/flypitch/_target/deps/mathlib/src/tactic/res
tate_axiom.lean:
par/Users/maruyama/flypitch/_target/deps/mathlib/src/tactic/simpa.lean:
parsing at
/Users/maruyama/flypitch/_target/deps/mathlib/src/tactic/restate_axiom.le
an:
```

.... 省略 ....

```
def bSet.CH : Π {𝔹 : Type u} [_inst_1 :
nontrivial_complete_boolean_algebra 𝔹], 𝔹 :=
λ {𝔹 : Type u} [_inst_1 : nontrivial_complete_boolean_algebra 𝔹],
  -⨆ (x : bSet 𝔹),
      Ord x ⊓
        ⨆ (y : bSet 𝔹),
          (λ (x y : bSet 𝔹), -larger_than x y) omega x ⊓ (λ (x y : bSet 𝔹), -
larger_than x y) x y ⊓
            (λ (x y : bSet 𝔹), injects_into x y) y (𝒫 omega)
def CH_f : sentence L_ZFC :=
∀'(Ord_f ⟹ substmax_bounded_formula at_most_f omega ⊔ at_most_f[P'
omega /0])
def 𝔹_cohen : Type :=
regular_opens (set (type ℵ₂ × ℕ))
def collapse_algebra.𝔹_collapse : Type u :=
collapse_algebra (type ℵ₁) (type (powerset omega))
propext
classical.choice
quot.sound
maruyama@MacBook-2 flypitch %
```

lean --trust=0 ./src/summary.lean
証明の実行

structure fol.Language : Type (u+1)
fields:
fol.Language.functions : Language → ℕ → Type u
fol.Language.relations : Language → ℕ → Type u
inductive fol.preterm : Language → ℕ → Type u
constructors:
fol.preterm.var : Π {L : Language}, ℕ → preterm L 0
fol.preterm.func : Π {L : Language} {l : ℕ}, L.functions l → preterm L l
fol.preterm.app : Π {L : Language} {l : ℕ}, preterm L (l + 1) → preterm L 0 → preterm L l
inductive fol.preformula : Language → ℕ → Type u
constructors:
fol.preformula.falsum : Π {L : Language}, preformula L 0
fol.preformula.equal : Π {L : Language}, term L → term L → preformula L 0
fol.preformula.rel : Π {L : Language} {l : ℕ}, L.relations l → preformula L l
fol.preformula.apprel : Π {L : Language} {l : ℕ}, preformula L (l + 1) → term L → preformula L l

fol.preformula.imp : Π {L : Language}, preformula L 0 → preformula L 0 → preformula L 0

fol.preformula.all : Π {L : Language}, preformula L 0 → preformula L 0

@[reducible]

def fol.term : Language → Type u :=

λ (L : Language), preterm L 0

@[reducible]

def fol.formula : Language → Type u :=

λ (L : Language), preformula L 0

@[reducible]

def fol.sentence : Language → Type u :=

λ (L : Language), presentence L 0

theorem fol.soundness : ∀ {L : Language} {T : Theory L} {A : sentence L}, T ⊢ A → T ⊨ A :=

λ {L : Language} {T : Theory L} {A : sentence L} (H : T ⊢ A), ssatisfied_of_satisfied (formula_soundness H)

inductive fol.prf : Π {L : Language}, set (formula L) → formula L → Type u

constructors:

fol.prf.axm : Π {L : Language} {Γ : set (formula L)} {A : formula L}, A ∈ Γ → Γ ⊢ A

fol.prf.impI : Π {L : Language} {Γ : set (formula L)} {A B : formula L}, insert A Γ ⊢ B → Γ ⊢ A ⟹ B

fol.prf.impE : Π {L : Language} {Γ : set (formula L)} (A : preformula L 0) {B : preformula L 0},
  Γ ⊢ A ⟹ B → Γ ⊢ A → Γ ⊢ B

fol.prf.falsumE : Π {L : Language} {Γ : set (formula L)} {A : formula L}, insert ~A Γ ⊢ ⊥ → Γ ⊢ A

fol.prf.allI : Π {L : Language} {Γ : set (preformula L 0)} {A : formula L}, lift_formula1 '' Γ ⊢ A → Γ ⊢ ∀' A

fol.prf.allE$_2$: Π {L : Language} {Γ : set (formula L)} (A : preformula L 0) (t : term L), Γ ⊢ ∀' A → Γ ⊢ A[t // 0]

fol.prf.ref : Π {L : Language} (Γ : set (formula L)) (t : term L), Γ ⊢ t ≃ t

fol.prf.subst$_2$: Π {L : Language} {Γ : set (formula L)} (s t : term L) (f : preformula L 0),
  Γ ⊢ s ≃ t → Γ ⊢ f[s // 0] → Γ ⊢ f[t // 0]

def fol.provable : Π {L : Language}, set (formula L) → formula L → Prop :=

λ {L : Language} (T : set (formula L)) (f : formula L), nonempty (T ⊢ f)

def fol.is_consistent : Π {L : Language}, Theory L → Prop :=

λ {L : Language} (T : Theory L), ¬T ⊢' ⊥

inductive pSet : Type (u+1)

constructors:

pSet.mk : Π (α : Type u), (α → pSet) → pSet

inductive bSet : Π (𝔹 : Type u) [_inst_1 : complete_boolean_algebra 𝔹], Type (u+1)

```
constructors:
bSet.mk : Π {𝔹 : Type u} [_inst_1 : complete_boolean_algebra 𝔹] (α :
Type u),
  (α → bSet 𝔹) → (α → 𝔹) → bSet 𝔹
def L_ZFC : Language :=
{functions := ZFC_func, relations := ZFC_rel}
def ZFC : Theory L_ZFC :=
{axiom_of_emptyset,
    axiom_of_ordered_pairs,
    axiom_of_extensionality,
    axiom_of_union,
    axiom_of_powerset,
    axiom_of_infinity,
    axiom_of_regularity,
    zorns_lemma} ∪
  ⋃ (n : ℕ), axiom_of_collection '' set.univ
```

$$(\forall x1,\neg\ x1 \in \emptyset)$$

$$(\forall x1,(\forall x2,(\forall x3,(\forall x4,(((\mathrm{pair}(x1)(x2) = \mathrm{pair}(x3)(x4)) \Rightarrow ((x1 = x3) \wedge (x2 = x4))) \wedge (((x1 = x3) \Rightarrow \neg\ (x2 = x4)) \vee (\mathrm{pair}(x1)(x2) = \mathrm{pair}(x3)(x4)))))))$$

$$(\forall x1,(\forall x2,((\forall x3,((x3 \in x1 \Rightarrow x3 \in x2) \wedge (x3 \in x2 \Rightarrow x3 \in x1))) \Rightarrow (x1 = x2))))$$

$$(\forall x1,(\forall x2,((x2 \in \cup(x1) \Rightarrow \exists x3,(x3 \in x1 \wedge x2 \in x3)) \wedge ((\forall x3,((x3 \in x1 \Rightarrow \neg\ x2 \in x3) \vee \bot)) \vee x2 \in \cup(x1)))))$$

$$(\forall x1,(\forall x2,((x2 \in \mathcal{P}(x1) \Rightarrow (\forall x3,(x3 \in x2 \Rightarrow x3 \in x1))) \wedge ((\forall x3,(x3 \in x2 \Rightarrow x3 \in x1)) \Rightarrow x2 \in \mathcal{P}(x1)))))$$

$$((\emptyset \in \omega \wedge (\forall x1,(x1 \in \omega \Rightarrow \exists x2,(x2 \in \omega \wedge x1 \in x2)))) \wedge (\exists x1,((((\forall x2,(x2 \in x1 \Rightarrow (\forall x3,(x3 \in x1 \Rightarrow (((x2 = x3) \vee x2 \in x3) \vee x3 \in x2))))) \wedge (\forall x2,((\forall x3,(x3 \in x2 \Rightarrow x3 \in x1)) \Rightarrow ((x2 = \emptyset) \vee \exists x3,(x3 \in x2 \wedge (\forall x4,(x4 \in x2 \Rightarrow \neg\ x4 \in x3))))))) \wedge (\forall x2,(x2 \in x1 \Rightarrow (\forall x3,(x3 \in x2 \Rightarrow x3 \in x1))))) \wedge (\omega = x1)) \wedge (\forall x1,((((\forall x2,(x2 \in x1 \Rightarrow (\forall x3,(x3 \in x1 \Rightarrow (((x2 = x3) \vee x2 \in x3) \vee x3 \in x2))))) \Rightarrow \neg\ (\forall x2,((\forall x3,(x3 \in x2 \Rightarrow x3 \in x1)) \Rightarrow ((x2 = \emptyset) \vee \exists x3,(x3 \in x2 \wedge (\forall x4,(x4 \in x2 \Rightarrow \neg\ x4 \in x3))))))) \vee \neg\ (\forall x2,(x2 \in x1 \Rightarrow (\forall x3,(x3 \in x2 \Rightarrow x3 \in x1))))) \vee ((\emptyset \in x1 \Rightarrow \neg\ (\forall x2,(x2 \in x1 \Rightarrow \exists x3,(x3 \in x1 \wedge x2 \in x3)))) \vee (\forall x2,(x2 \in \omega \Rightarrow x2 \in x1)))))))$$

$$(\forall x1,((x1 = \emptyset) \vee \exists x2,(x2 \in x1 \wedge (\forall x3,(x3 \in x1 \Longrightarrow \neg\ x3 \in x2)))))$$

$$(\forall x1,((x1 = \emptyset) \vee ((\forall x2,(((\forall x3,(x3 \in x2 \Longrightarrow x3 \in x1)) \Longrightarrow \neg\ (\forall x3,(\forall x4,((x3 \in x2 \Longrightarrow \neg\ x4 \in x2) \vee ((\forall x5,(x5 \in x3 \Longrightarrow x5 \in x4)) \vee (\forall x5,(x5 \in x4 \Longrightarrow x5 \in x3))))))) \vee \cup(x2) \in x1)) \Longrightarrow \exists x2,(x2 \in x1 \wedge (\forall x3,(x3 \in x1 \Longrightarrow ((\forall x4,(x4 \in x2 \Longrightarrow x4 \in x3)) \Longrightarrow (x2 = x3))))))))$$

```
def bSet.CH : Π {𝔹 : Type u} [_inst_1 :
nontrivial_complete_boolean_algebra 𝔹], 𝔹 :=
λ {𝔹 : Type u} [_inst_1 : nontrivial_complete_boolean_algebra 𝔹],
  -⨆ (x : bSet 𝔹),
      Ord x ⊓
        ⨆ (y : bSet 𝔹),
          (λ (x y : bSet 𝔹), -larger_than x y) omega x ⊓ (λ (x y : bSet 𝔹), -
larger_than x y) x y ⊓
            (λ (x y : bSet 𝔹), injects_into x y) y (𝒫 omega)
```

```
def CH_f : sentence L_ZFC :=
∀'(Ord_f ⟹ substmax_bounded_formula at_most_f omega ⊔ at_most_f[P' omega /0])
def 𝔹_cohen : Type :=
regular_opens (set (type ℵ₂ × ℕ))
def collapse_algebra.𝔹_collapse : Type u :=
collapse_algebra (type ℵ₁) (type (powerset omega))
propext
classical.choice
quot.sound
maruyama@MacBook-2 flypitch %
maruyama@MacBook-2 flypitch %
```

Feit–Thompson定理の形式的証明

Feit–Thompson定理の形式的証明

2012年、Georges Gonthierは、Coqを用いて、群論のFeit–Thompson 定理の形式的証明に成功する。

この定理は、1962-1963年にWalter Feit と John Griggs Thompson によって証明された、群論の基本的な定理の一つである。

この定理は、「奇数位数の有限単純群は可解である」ことを主張し(「奇数位数定理 = “Odd Order Theorem” とも呼ばれる」)、有限単純群の分類問題で重要な役割を果たしている。

有限単純群の分類問題は
非常に膨大な証明である

「有限単純群の分類問題」は、Daniel Gorensteinをリーダーとする数学者の集団によって解決され、20世紀の数学の大きな成果の一つと呼ばれている。

ただ、その「証明」は、100人以上の数学者が、50年のあいだに数百の論文誌に発表した、膨大な量の「証明」の総和である。そのページ数は一万ページを超えると言われている。

Georges Gonthier

Gonthierの ”Feit–Thompson theorem” の型式的証明も、膨大なものである。それは、15,000の定義、4,300の定理、17万行のソースからなる。この証明を、彼はチーム作業で、6年かけて完成させた。

https://github.com/math-comp/odd-order

“A Machine-Checked Proof of the Odd Order Theorem”
https://hal.inria.fr/hal-00816699/document

opam install coq-mathcomp-odd-order

```
<><> Processing actions <><><><><><><><><><><><><><><><><><><><><><><><><><><><>
⬇ retrieved coq-mathcomp-field.1.14.0  (https://github.com/math-comp/math-comp/archive/mathcomp-1.14.0.tar.gz)
⬇ retrieved coq-mathcomp-fingroup.1.14.0  (cached)
⬇ retrieved coq-mathcomp-character.1.14.0  (https://github.com/math-comp/math-comp/archive/mathcomp-1.14.0.tar.gz)
⬇ retrieved coq-mathcomp-solvable.1.14.0  (cached)
⬇ retrieved coq-mathcomp-algebra.1.14.0  (https://github.com/math-comp/math-comp/archive/mathcomp-1.14.0.tar.gz)
⬇ retrieved coq-mathcomp-ssreflect.1.14.0  (cached)
⬇ retrieved coq-mathcomp-odd-order.1.13.0  (https://github.com/math-comp/odd-order/archive/mathcomp-odd-order.1.13.0.tar.gz)
Processing  8/21: [coq-mathcomp-ssreflect: make 3]
∗ installed coq-mathcomp-ssreflect.1.14.0
Processing 10/21: [coq-mathcomp-fingroup: make 3]
∗ installed coq-mathcomp-fingroup.1.14.0
∗ installed coq-mathcomp-algebra.1.14.0
∗ installed coq-mathcomp-solvable.1.14.0
∗ installed coq-mathcomp-field.1.14.0
∗ installed coq-mathcomp-character.1.14.0
∗ installed coq-mathcomp-odd-order.1.13.0
Done.
```

Theorem Feit_Thompson

PFsection14.v

theories > PFsection14.v

```
1263    Theorem Feit_Thompson (gT : finGroupType) (G : {group gT}) :
1264      odd #|G| -> solvable G.
1265    Proof. exact: (minSimpleOdd_ind no_minSimple_odd_group). Qed.
1266
1267    Theorem simple_odd_group_prime (gT : finGroupType) (G : {group gT}) :
1268      odd #|G| -> simple G -> prime #|G|.
1269    Proof. exact: (minSimpleOdd_prime no_minSimple_odd_group). Qed.
1270
```

Theorem stripped_Odd_Order

stripped_odd_order_theorem.v

~/odd-order/theories/
stripped_odd_order_theorem.v

```
204
205    Theorem stripped_Odd_Order T mul one inv (G : T -> Type) (n : natural) :
206        group_axioms T mul one inv -> group T mul one inv G ->
207        finite_of_order T G n -> odd n ->
208      solvable_group T mul one inv G.
209    Proof. exact (InternalSkepticOddOrderProof.main T mul one inv G n). Qed.
210
```

証明のチェック
https://github.com/math-comp/odd-order

coq-mathcomp-odd-order.opam

```
21  The formal proof of the Feit-Thompson theorem.
22
23  From mathcomp Require Import all_ssreflect all_fingroup all_solvable PFsection14.
24
25  Check Feit_Thompson.
26      : forall (gT : finGroupType) (G : {group gT}), odd #|G| -> solvable G
27
28  From mathcomp Require Import all_ssreflect all_fingroup
29                               all_solvable stripped_odd_order_theorem.
30
31  Check stripped_Odd_Order.
32      : forall (T : Type) (mul : T -> T -> T) (one : T) (inv : T -> T)
33               (G : T -> Type) (n : natural),
34          group_axioms T mul one inv ->
35          group T mul one inv G ->
36          finite_of_order T G n -> odd n -> solvable_group T mul one inv G"""
37
```

Gonthierと「四色問題」

1974年、イリノイ大学のKenneth AppelとWolfgang Hakenは、当時の最先端のコンピュータを使って、「四色問題」を解いた。

コンピュータの稼働時間は、1200時間に及び、夜間の空き時間しか利用が認められなかったので半年がかりの計算だったと言う。

2004年、Gonthierは、Coqを使って「四色問題」を解いた。

"A computer-checked proof of the Four Colour Theorem"

http://www2.tcs.ifi.lmu.de/~abel/lehre/WS07-08/CAFR/4colproof.pdf

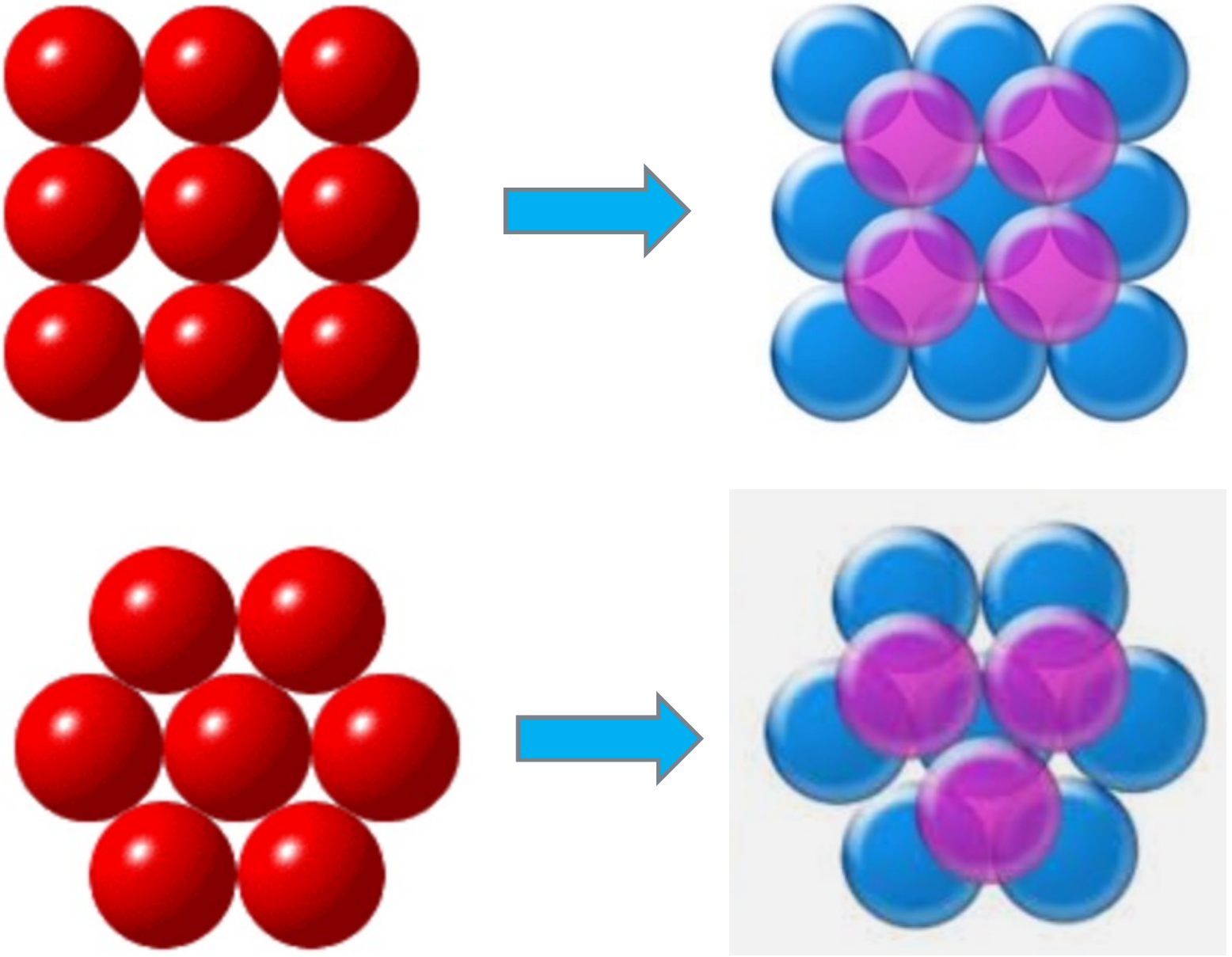
ケプラー予想の証明

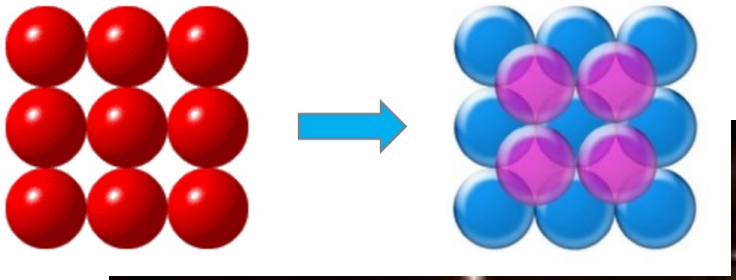
ケプラー予想

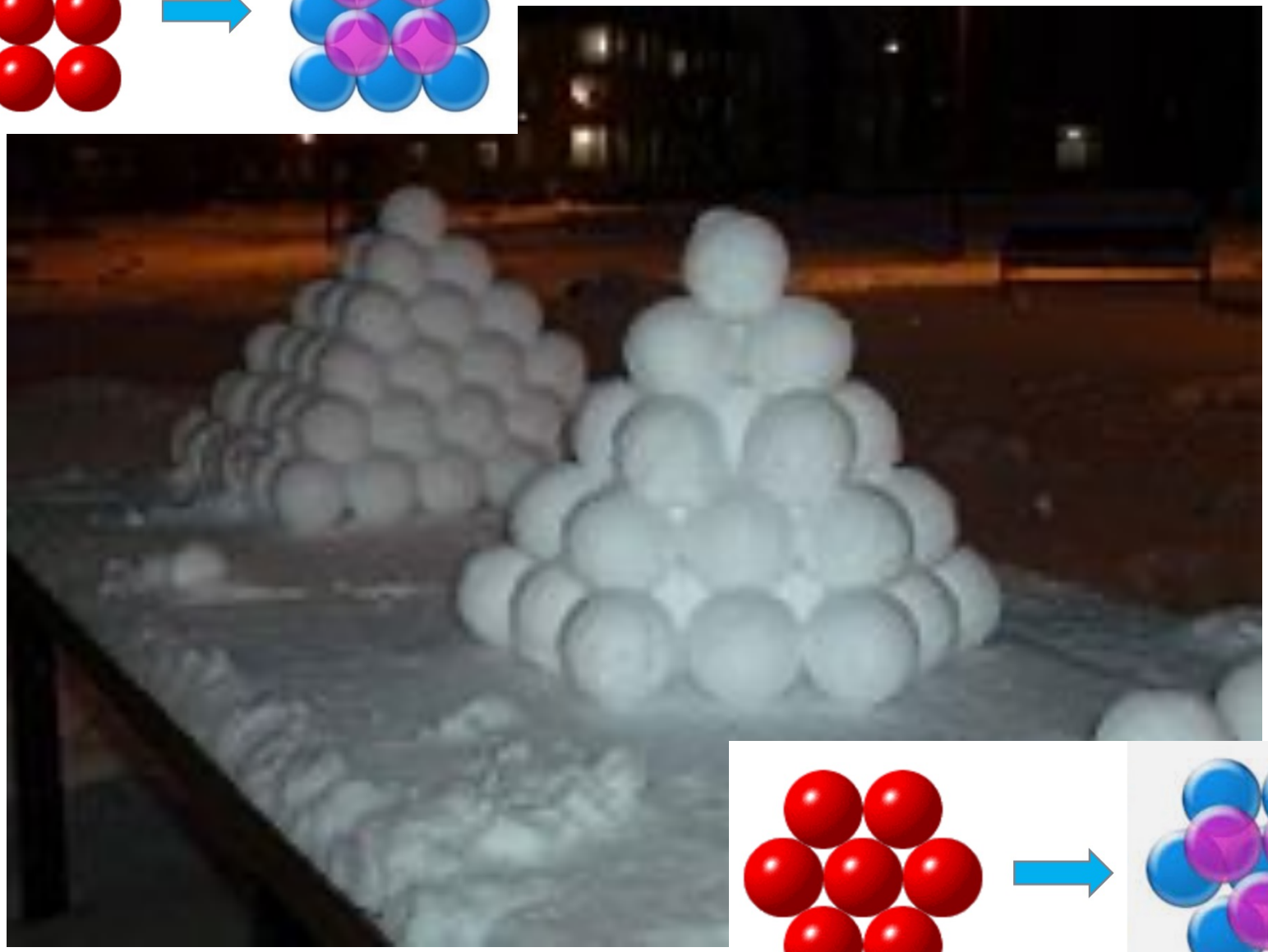
「ケプラー予想」は、入れ物の箱に同じ大きさのボールを、どのくらいたくさん詰め込むことができるかという問題に関するで予想である。

何も考えずに、箱の上からボールを流し込むと、充填率（詰め込んだボール全体の体積を、容器の箱の体積で割ったもの）は、平均すると 65%程度になるという。箱の35%は、すきまになるということだ。

もっと、びっしりと、ボールを詰め込みことはできないか？
ケプラーは、もっとボールを詰め込むことができる二つの方法を考えた。







ケプラー予想、300年解かれず
ヒルベルトの18番目の問題になる

二つの充填率は、ともに、74.05%で等しい。

正確にいうと、それは $\frac{\pi}{3\sqrt{2}}$=0.740480489... となる。

1611年、ケプラーは、どんなボールの詰め方をしても、この数字を超えるほど密にボールを詰め込むことはできないだろうと予想した。これを「ケプラー予想」という。ケプラー自身は、この問題に証明を与えることはできなかった。

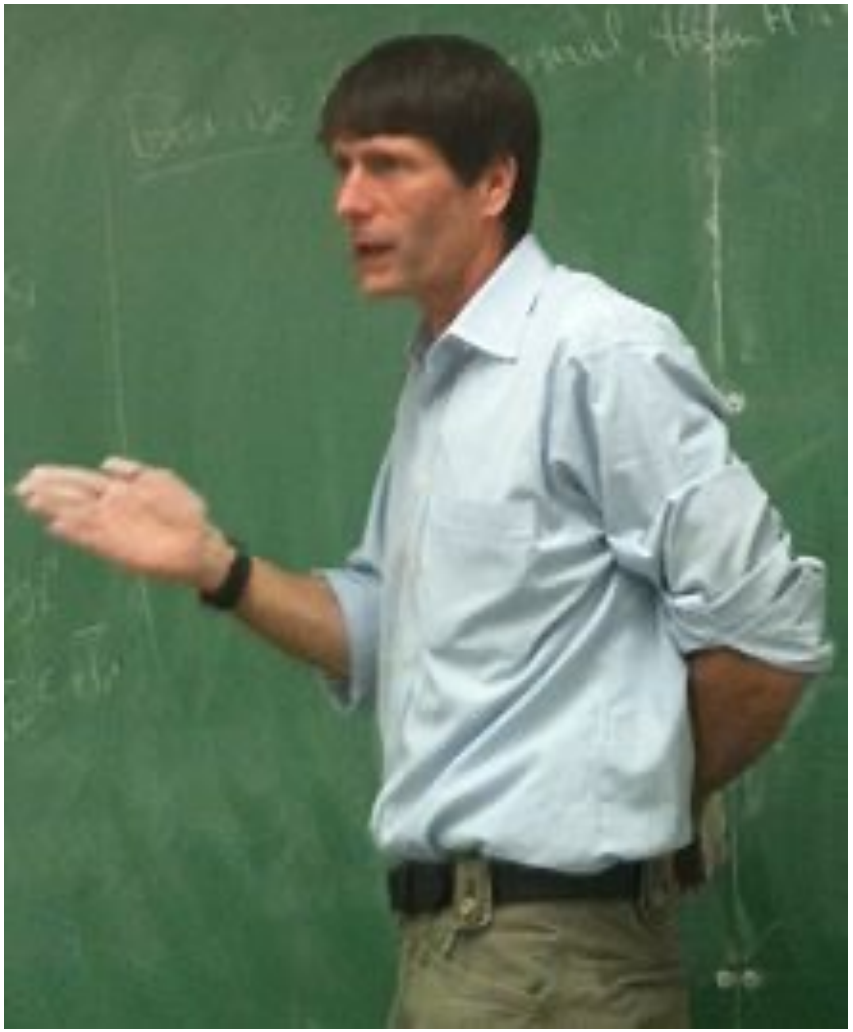
1900年のヒルベルトの20世紀の数学が解くべき23の問題の 18番目にこの問題は、取り上げられている。

ケプラー予想の400年

1611年

Johannes Kepler

2014年



Thomas Hales

Hales ケプラー予想を解く

1992年、Thomas HalesはSamuel Fergusonとともに、可能なボールの配置について総当たりで、線形計画法の手法で充填率の最大値を求めることで、ケプラー予想が解けると予想する。

1998年、Halesらは、250ページの論文と2GByteのプログラムとデータを提出し、ケプラー予想を解いたと主張する。ただ論文の査読者は「99%正しい」としたものの、1%を留保した。

2003年、Halesはケプラー予想のコンピュータによる完全に形式的証明をめざす Flyspec プロジェクトを立ち上げる。

2015年、Halesと協力者21名は、論文 "A formal proof of the Kepler conjecture"をarXivに投稿し、ケプラー予想の解決を宣言する。

2017年には、彼らの証明は、学会に受理される。

/flyspec/text_formalization/general/audit_formal_proof.hl

audit_formal_proof.hl

~/flyspeck/text_formalization/general/
audit_formal_proof.hl

audit_formal_proof.hl

```
36
37  (* The definition of packing *)
38
39  let chk_packing_def =
40    chk_thm(Sphere.packing_lt,
41     `packing (V:real^3 -> bool) =
42        (!u:real^3 v:real^3. (u IN V) /\ (v IN V) /\ (dist( u, v) < &2) ==>
43     (u = v))`);;
44
45  (* The definition of the Kepler conjecture *)
46
47  let chk_kepler_conjecture_def = chk_thm
48    (The_main_statement.the_kepler_conjecture_def,
49     `the_kepler_conjecture <=>
50       (!V. packing V
51                ==> (?c. !r. &1 <= r
52                             ==> &(CARD(V INTER ball(vec 0,r))) <=
53                                 pi * r pow 3 / sqrt(&18) + c * r pow 2))`
54    );;
55
```


50

第四部

形式的証明の背景と変化の意味を考える

数学の基礎とコンピュータの利用

数学の証明を記述するスタイルは、今、大きく変わろうとしている。これは、21世紀の数学を特徴づけるものになるだろう。

第四部では、こうした流れを切り拓いた Voevodskyの数学の基礎についてのUnivalent foundationsという考えを紹介する。

彼のプロジェクト UniMathは、コンピュータで検証可能なスタイルで数学を記述しようとする、初めての試みである。
https://github.com/UniMath/UniMath

コンピュータ自身が、「推論能力」を持つということ

Homotopy Type Theoryについては、別のシリーズを予定している。

「カテゴリー論入門」セミナーについて

https://www.marulabo.net/category-theory/

今回のセミナーでは、Open-AI Theorem Proverのようないわゆる AIがコンピュータに「推論能力」を与えるのではなく、そもそもコンピュータ自身が、「推論能力」を持っているという立場で議論を進めたい。

第四部
形式的証明の背景と変化の意味を考える

なぜ、証明にコンピュータを使うのか

数学的知の「累積性」

Univalent Foundation

広がる形式的証明

なぜ、証明にコンピュータを使うのか

なぜ、コンピュータを使い始めたのか？

数学者が証明問題を解くのに、コンピュータを利用し始めていることを、いくつかの例で紹介してきた。

では、なぜ、数学者はコンピュータを使い始めたのだろうか？それには、理由があるように思う。ここでは、それを考えてみよう。

人の手にはあまるが、コンピュータなら解ける巨大なサイズの問題が存在すること

数学者が、問題を解くのにコンピュータを利用する必要性で、一番分かりやすいのは、人の手にはあまるが、コンピュータなら解ける巨大なサイズの問題が存在することである。

一般的な解法は見当たらなくても、場合の数が有限なら、すべての場合をしらみつぶしに総当たりでチェックして問題を解くことができる。

数学の証明へのコンピュータ利用の先駆けとなった、1970年代の「四色問題」の証明はこうしたタイプの問題だった。先に見た、Halesの「ケプラー予想」の証明も、同じタイプである。

人の手にはあまるが、コンピュータなら解ける巨大なサイズの問題が存在すること

ただ、留意すべきことが二つある。一つは、数学の問題がすべてこうしたタイプである訳ではないということである。

もう一つは、場合の数が有限だとしても、総当たりが実際にコンピュータで可能とは限らないということである。3-SAT問題がむずかしいのは、しらみつぶしには、指数関数的な計算時間が必要となるからである。

ここで、このタイプの問題は、計算複雑性理論のNP完全問題と接点を持つことになる。運が良ければ（場合の数が小さければ）、しらみつぶしで問題は解ける。

証明が正しいことの検証の必要性

コンピュータがある問題の証明のために、膨大な計算を行ったとしても、そのことだけでは、その証明が正しいと主張することはできない。その計算が証明の正しさを示すことを、自ら検証する必要がある。

ヘイルズが、flyspec プロジェクトを立ち上げたのは、まさにそのためである。コンピュータ自身に、証明が正しいことの検証の能力を持たせることは、本質的に重要である。

その意味では、70年代のアッペルとハーケンによる「四色問題」の証明と、21世紀のゴンティエによる「四色問題」の証明とは、違うものである。

証明が正しいことの検証の必要性

重要なことは、証明が正しいことの検証の必要性は、なにもコンピュータによる巨大な計算を必要とする証明に限られる訳ではないということである。

コンピュータを使う使わないを問わず、証明にはそれが正しいことの検証が求められる。それは、従来はアカデミーの論文採択のチェック機能に主要に支えられてきた。ただ、そのアカデミーの機能を支えているのは、人間である。

証明の正しさの検証が、人の手にあまることはないのだろうか？

Grigory Perelman

Grigory Perelmanは、2002年から2003年にかけて arXiv に投稿した論文で、「三次元のポアンカレ予想」を解いたと主張した。

arXivは、学会の論文誌とは異なり、掲載の可否を決める査読が基本的にないので、発表の時点では、彼の論文以外に、彼の証明が正しいという裏付けはなかった。

彼の証明の検証は、複数の数学者のグループで取り組まれ、最終的に彼の証明の正しさが「検証」されたのは、2006年のことだった。

証明から誤りを排除する必要性

ただ、「証明の正しさ」を検証するために、コンピュータの利用が必要なだけではない。

もっと基本的なところで、コンピュータの利用が必要なのである。

なぜなら、数学者は、しばしば「間違った証明」を与えるからである。

誤った「証明」

「フェルマーの定理」は、最終的には、1995年のAndrew Wilesの論文によって証明されたが、フェルマー自身は、この定理を証明出来たと信じていた。それが、誤った「証明」であったのは確かである。

Wiles自身も、95年の論文以前に一度、「証明した」とする発表を行うが、誤りが見つかり、それを撤回している。

世紀の難問である「リーマン予想」は、何度か「証明」が発表されている。今までのところ、それは誤って「証明」だった。

数学的知の「累積性」

数学的知の「累積性」

数学的知には、人間の認識によって獲得されたものとしては、独特の性質がある。

それは、いったん数学的に「真」であることが証明された定理は、時代が変わっても場所が変わっても、ずっと「真」のままであり続けることだ。

そうして獲得された知は、個人というよりは人類の知として蓄積されていく。それを数学的知の「累積性」という。

「巨人の肩の上の小人」

「フェルマーの定理」を証明したワイルズは、自分を「巨人の肩の上の小人」に喩えたのだが、それは彼の謙遜であると同時に、数学的知の体系が「累積的」であることを彼がはっきりと自覚していることを示している。

マルゼミのセミナー「認識の認識」
https://www.marulabo.net/docs/philosophy02/
で紹介した、GrzegorczykやKripkeのモデルは、いずれも知の累積性に基づいた認識モデルである。

「形式的証明」が求められる最大の理由

先に、数学者がコンピュータによる形式的証明を受容する動きが広がっている理由をいくつか見てきたのだが、最後にあげた「数学者は、しばしば「間違った証明」を与えるから」というのが、結局は一番大きな理由になるだろうと僕は考えている。

ただ、数学者も一般の人も、あまりそのことを深刻に考えていないようにも見える。

数学者の誤りの問題を、数学の基礎である累積性への脅威として捉えたのは、Voevodsky である。

Vladimir Voevodsky

2000年頃、彼は1993年に自分が発表した論文の重要な補題が間違っていたことに気づく。でも、その頃には、その論文は広く出回っていて、多くの数学者がその証明を「信じて」いた。彼が、その間違った補題なしでも、論文の結論が正しいことを証明できたのは、2006年になってからだった。

別のこともあった。1998年に共著で彼が発表した論文の証明に対して、「正しくない」という批判が出される。結論的には、彼は、正しかったのだが、彼が、最終的に、自分が正しいことを確信できたのは、2013年になってからだった。

Univalent Foundation

Voevodskyが危惧したこと

数学は、累積的（accumulative）な性格を持つので、そこに誤りが紛れ込むと、それも、累積する危険がある。

And since mathematics is a very deep science, in the sense that the results of one article usually depend on the results of many and many previous articles, this accumulation of errors for mathematics is very dangerous.

証明へのコンピュータの利用

そうした数学への誤りの混入を避ける唯一の道は、数学の証明を、コンピュータでチェックできるプログラムの形にすることである。

And it soon became clear that the only long-term solution was somehow to make it possible for me to use computers to verify my abstract, logical, and mathematical constructions.

しかし、当初は、そうした主張に耳を傾ける数学者は、ほとんどいなかった。

Univalent Foundation

しかし、Voevodskyが、Homotopy Type Theory に基づく、新しい数学の基礎づけ Univalent Foundation を発表すると、状況は大きく変わった。

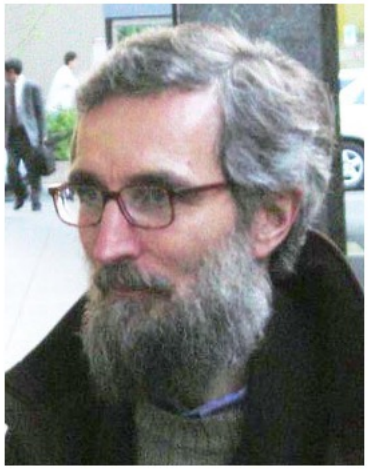
Today, only a few years later, computer verification of proofs and of mathematical reasoning in general looks completely practical to many people who work on univalent foundations and homotopy type theory.

Homotopy Theory

William Lawvere

Alexander Grothendieck

Type Theory



Per Martin-Löf

Homotopy Type Theory

Vladimir Voevodsky

Univalent Axiom

Univalent Foundation

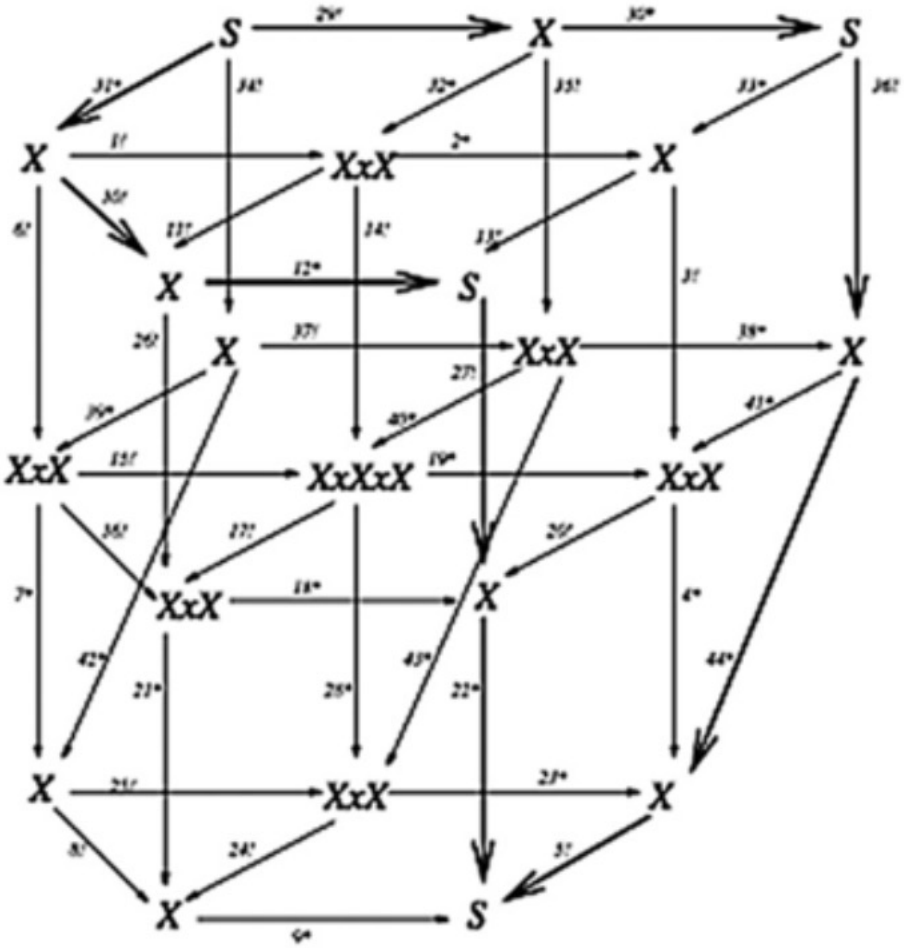
Univalent Foundationは
いかにしてうまれたのか？

Univalent Foundationの起源と動機については、Voevodsky の次の論文を参照されたい。

“The Origins and Motivations of Univalent Foundations”

https://www.ias.edu/ideas/2014/voevodsky-origins

この論文の副題は、”A Personal Mission to Develop Computer Proof Verification to Avoid Mathematical Mistakes”
である。



UniMath

特筆すべきは、Voevodsky は、その新しい数学の基礎づけ Univalent Foundationを、彼の主張に忠実に、論文ではなく GitHubでプログラムの形で公開していることである。

そのライブラリーをUniMathという。
https://github.com/UniMath/UniMath
https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/2016_09_22_HLF_Heidelberg.pdf

こうしたスタイルは、21世紀の数学に、大きな影響を与えるだろう。


50

Voevodsky とのインタビュー

https://baaltii1.livejournal.com/198675.html

Baezによる英訳
https://johncarlosbaez.wordpress.com/2017/10/06/vladimir-voevodsky-1966-2017/

数学の二つの危機
第一：純粋数学と応用数学の分離

I realized that mathematics is on the verge of a crisis, or rather, two crises.

The first is connected with the separation of “pure” and applied mathematics. It is clear that sooner or later there will be a question about why society should pay money to people who are engaged in things that do not have any practical applications.

https://baaltii1.livejournal.com/198675.html

Baezによる英訳 https://johncarlosbaez.wordpress.com/2017/10/06/vladimir-voevodsky-1966-2017/

数学の二つの危機
第二：検出されないエラーの蓄積

The second, less obvious, is connected with the complication of pure mathematics, which leads to the fact that, sooner or later, the articles will become too complicated for detailed verification and the process of accumulating undetected errors will begin. And since mathematics is a very deep science, in the sense that the results of one article usually depend on the results of many and many previous articles, this accumulation of errors for mathematics is very dangerous.

So, I decided, you need to try to do something that will help prevent these crises. For the first crisis, this meant that it was necessary to find an applied problem that required for its solution the methods of pure mathematics developed in recent years or even decades.

I think it was at this moment that I largely stopped doing what is called “curiosity-driven research” and started to think seriously about the future. I didn’t have the tools to explore the areas where curiosity was leading me and the areas that I considered to be of value and of interest and of beauty.

唯一の解は、証明の検証に
コンピュータを使うことである

So I started to look into what I could do to create such tools. And it soon became clear that the only long-term solution was somehow to make it possible for me to use computers to verify my abstract, logical, and mathematical constructions. The software for doing this has been in development since the sixties. At the time, when I started to look for a practical proof assistant around 2000, I could not find any. There were several groups developing such systems, but none of them was in any way appropriate for the kind of mathematics for which I needed a system.

When I first started to explore the possibility, computer proof verification was almost a forbidden subject among mathematicians. A conversation about the need for computer proof assistants would invariably drift to Gödel's incompleteness theorem (which has nothing to do with the actual problem) or to one or two cases of verification of already existing proofs, which were used only to demonstrate how impractical the whole idea was. Among the very few mathematicians who persisted in trying to advance the field of computer verification in mathematics during this time were Tom Hales and Carlos Simpson.

Today, only a few years later, computer verification of proofs and of mathematical reasoning in general looks completely practical to many people who work on univalent foundations and homotopy type theory.