
コンピュータと数学
TuringからVeovodskyまで

Agenda

コンピュータと数学　TuringからVeovodskyまで

Part 1

機械は考えることができるか？

Part 2

「計算 ＝ 証明 ＝ プログラム」という認識の発展

Part 3

コンピュータによる証明

Part 1

機械は考えることができるか？

Part 1 Agenda
機械は考えることができるか？

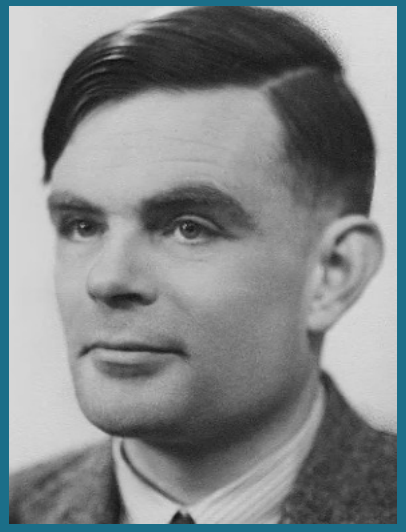
はじめに

Turingが考えたこと

Veovodskyが考えたこと

コンピュータと数学 1

TuringからVeovodskyまで



はじめに

大規模言語モデルと生成AIの成功
AI技術の次の未来を展望する

「大規模言語モデル」ベースの現代の生成AI技術は、ネット上の膨大なテキストの集積を学習して、そこから、人間の「言語能力」と呼ばれているものの基本的部分 --「意味の理解」「新しい文の生成」「対話形のコミュニケーション」、「文字として蓄積された情報の活用」等 -- をコンピュータ上で実現することに成功しました。

それは画期的なものです。この連続セミナーは、歴史的な成功をおさめたAI技術の次の未来を展望することを目的としています。

それなら、「AIの未来」のようなタイトルの方が良かったと思われるかもしれません。

ただ、一見するとAIとの関係がはっきりしない、一般的な「コンピュータと数学」というタイトルを選んだのには理由があります。それは、AIの未来についてのつぎのような僕のビジョンに関係しています。

AIの未来についての三つのビジョン

第一。僕は、未来の（あるいは、近未来なのかも）AIの発展の課題の焦点の一つは、人間の二つの基本的能力である「言語能力」と「数学的＝論理的能力」の「統合」の課題だと考えています。

第二。ただ、「大規模言語モデル」は人間の「言語能力」の理解に画期をもたらしたのですが、ただ、その方法の素朴な延長では、AIの数学＝論理的能力の飛躍は望めないだろうと考えています。

それなら、「AIと数学」というタイトルがいいのではという考えもあると思います。でも、そうしたタイトルを選びませんでした。

一つには、大規模言語モデルベースの生成AI技術の華々しい成功の中で、ほとんどの人が「AI ＝ 生成AI」と思っている現状では、先の「第二」の論点は、なかなか伝えるのが難しいと感じたからです。

コンピュータは、それ自身で
「数学的＝論理的能力」を持っている

もう一つには、僕がこのセミナーで伝えたいAIの未来について、もう一つのメッセージがあるからです。

第三。そもそも、コンピュータは、それ自身で「数学的＝論理的能力」を持っているという考えかたがあるということです。

こうした考えを進めれば、わざわざ、「大規模言語モデル」の上に数学的＝論理的能力を構築しようとするのは、コンピュータが本来持つ力を過少に評価しているのであって、それは余計な回り道だということになります。

僕は、こうした考えに魅力を感じています。

こうした考えは、新しいものではありません。それは、コンピュータの歴史と同じくらい古いものです。ですが、生成AIの登場によってその考えが古びるものではないと僕は考えています。

「AI」という言葉は、少し窮屈

今回のセミナーでは、「AI」と言うことばをタイトルから外して、「コンピュータ」ということばに置き換えてみました。それで、すこしは自由にAIの未来を考えることができると感じています。

要するに、「AIの未来」を、「AI」を主語とするのではなく、「コンピュータ」を主語として、「コンピュータの未来」というパースペクティブで考えてみようというのが、今回のセミナーの趣旨です。僕の問題意識からすると、その焦点は、コンピュータの数学的能力だということになります。

もっとも、今回のセミナーが、「数学の歴史の話」でも「生成AI批判」でもなく、歴史的な成功をおさめたAI技術の次の未来を展望することを目的としていることを説明するのに、これだけグダグダ語らねばならないというのは、どこかでボタンを掛け違えたのかもしれませんね。

ただ、このシリーズ少し続けてみようと思います。

今回のセミナーの構成

今回のセミナー「コンピュータと数学 1 -- TuringからVeovodskyまで」は、次のような構成をしています。

- Part 1 機械は考えることができるか？
  - ・Turingが考えたこと
  - ・Veovodskyが考えたこと
- Part 2「計算 = 証明 = プログラム」という認識の発展
  - ・Curry-Howard 対応
  - ・Martin-LofのDependent Type Theory
  - ・VeovodskyのHomotopy Type Theory
- Part 3 コンピュータによる数学的証明の実例

  「数学とコンピュータ 1」ページで展開中

TuringからVeovodskyまで

Turingは、「機械は考えることができるか？」という問題を、初めて提起しました。Veovodskyは、「すべての数学的証明はコンピュータ・プログラムの形をとるべきだ。」と主張し、それを実行しようとしました。二人のコンピュータ観には、大きな飛躍があります。

僕は、こうした飛躍をもたらしたものが、「型の理論」の中で成立した「計算 ＝ 証明 ＝ コンピュータ・プログラム」という認識であると考えています。「TuringからVeovodskyまで」というサブタイトルを持つこのセミナーでは、重点的に「型の理論」の発展の歴史を紹介しています。

最後のパートは、コンピュータ・プログラムによる数学的証明の実例を紹介しています。

連続セミナー「コンピュータと数学」の展開について

連続セミナー「コンピュータと数学」は、今後、次のような展開を予定しています。ご期待ください。

- 「コンピュータと数学 1 -- TuringからVeovdskyまで」
- 「コンピュータと数学 2 -- Turingマシンから量子コンピュータまで」
- 「コンピュータと数学 3 -- GATlab」
- 「コンピュータと数学 4 -- Agent Based Modelの数学」

当初、今回のセミナーで取り上げると予告していたGATlabについては、「AIの未来」を「コンピュータと数学」という切り口から考えようとするアプローチの背景を、もう少し丁寧に紹介してから話してみたいと思っています。急なコンテンツの変更、お詫びします。

Turingが考えたこと

今年は、Turingの論文の75周年

すこし、論点を変えましょう。

AIの時代を切り拓いたのは、今から1950年のTuringの論文だと思います。その画期的な論文 “Computing Machinery and Intelligence" の冒頭で、彼はこう語ります。

「機械は考える事が出来るか? (Can machines think?)」
という問題を考察することを提案する。

今なら、誰かが「AIは考えることができるか？」といっても、誰も驚かないし、そうした問題提起が大きな関心を集めることもないように思います。

もちろん、「生成AIで数学の問題が解ける」と主張しても、「生成AIで数学の問題は解けない」と主張しても、それが異端視されることはありません（多分）。

ただ、この論文が出された1950年ではそうではありませんでした。彼は、「機械は考えることができる」という主張が、「異端」とみなされるだろうことをはっきりと自覚していました。

人々の「機械の思考」についての考え方は大きく変わったのです。

Turingが偉大だと僕が思うのは、そうした変化が起こるだろうことをも、彼が正しく予見していたことです。

「『機械は考える事が出来るか?』という、最初に掲げた問題が、今では議論にも値しない程無意味なものである事は私も認めよう。しかし、それにもかかわらず、今世紀の終わりには人達の意見が大きく変化して、ついには、矛盾していると考えることなく『機械の思考』について語ることができるようになるであろうと私は信じている。」

変化は急激に進んだ

もっとも、こうした変化が起きるのには、Turingが予想した「今世紀の終わり」までの50年では足らず、さらに四半世紀、都合50+25＝75年の時間が必要でした。

といっても、かつては「議論にも値しない程無意味なもの」と思われていたものが、「ついには、矛盾していると考えることなく『機械の思考』について語る」ようになる現在への変化は、75年かけてゆっくりとつくられたものではありません。

「AI」についての関心の高まりは何度かの波がありました。ただ、ITの世界全体を巻き込み、さらに多くの人を巻き込んで進行した現在の変化は、実際には、「生成AI」が登場したこの1-2年の間に急激に進んだものだと思います。

この変化の急激さは、この変化が広く強いインパクトをもって受け止められたことを物語っています。

それは、本質的には、人間が生み出した「機械は考えることができる」という、歴史的と言っていい認識の「転換」です。この変化を知ったら、Turingは喜んだはずです。

同時に、この変化の急激さは、多くの人にとって「AI」についての理解を深める時間がほとんどなかったことを意味します。

僕が興味深いと思うのは、Turingの予言が経てきたこの75年は、「AI」にとってではなく、「コンピュータ」にとってみれば、自身の誕生から現在の成熟までの全生涯の長さとと、ほぼ一致すると言うことです。

我々は、コンピュータの歴史も、computer scienceの歴史も、比較的よく知っています。

「AIの未来」を、「AI」を主語とするのではなく、「コンピュータ」を主語として、「コンピュータの未来」というパースペクティブで考えてみようというのが、今回のセミナーの意図です。

そもそも、コンピュータは「数学的＝論理的能力」を持つという主張

今回のセミナーで紹介しようと思っているのは、そもそも、コンピュータは、それ自身で「数学的＝論理的能力」を持っているという考えかたがあるということです。

こうした考えを進めれば、わざわざ、「大規模言語モデル」の上に数学的＝論理的能力を構築しようとするのは、コンピュータが本来持つ力を過少に評価しているのであって、それは余計な回り道だということになります。

僕は、こうした考えに魅力を感じています。

こうした考え方の起源は、決して新しいものではありません。

Turingの論文が出る20年前の1930年代、もちろんコンピュータは存在さえしていなかったのですが、$Gödel$ の不完全性定理の発見に触発されて、「証明とは何か？」「計算とは何か？」という研究が始まります。

こうして、さまざまな「計算可能性」の理論が提案されます。
$Gödel$ の「帰納関数論」、Churchの「ラムダ計算」、Turingの「Turing Machine」 等々。

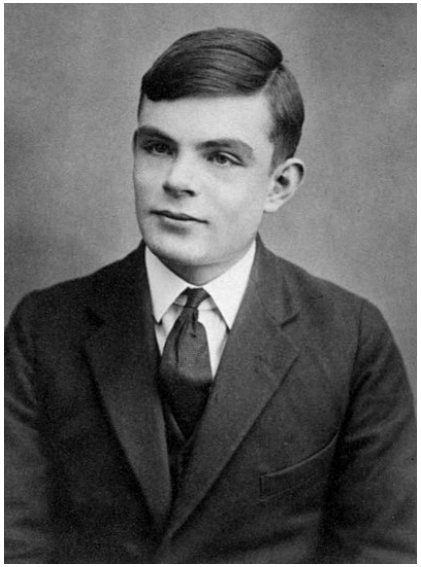
これらの時代の研究の白眉は、これらの見かけは全く異なる理論が全て同値であることが証明されたことです。

50年代には、「計算可能性」と「証明可能性」が同一の概念であることも、広く認識されることになります。（「Church-Turingのテーゼ」）

様々な「計算可能性」へのアプローチと
その同値性の認識

Kurt Gödel
1906-1978

Alonzo Church
1903-1995



Alan Turing
1912-1954

帰納関数論 ＝ ラムダ計算 ＝ チューリングマシン

そうした認識は、数学的なものの特徴は、対応するTuring Machine の特徴として把握することができるという認識に他なりません。

コンピュータの利用が拡大し、「型の理論」が「従属型の理論」へと進化した 1970年代には、コンピュータのプログラムの役割が数学的証明に他ならないことが見出されます。

21世紀に入って、新しい型の理論 Homotopy Type Theory (HoTT)によってカテゴリー論を新しい段階に引き上げたVoevodskyは、「すべての数学理論は、コンピュータ・プログラムの形で記述されるべきだ」と考え、それを実行しようとしました。

残念ながら、この壮大な計画は、早すぎる彼の死によって、中断されました。

Voevodskyが考えたこと

Veovodskyって誰？

Voevodskyは、Milner予想、Bloch-Kato予想を解くなど、代数幾何でグロタンディックが進もうとした道で、大きな業績を残した数学者です。

2002年、26歳の時、彼はこうした業績でフィールズ賞を受賞します。

（北京で行われた、授与式の様子。）

2017年、彼は動脈瘤で突然亡くなります。51歳でした。

多くの数学者が、早すぎる彼の死を悼みました。
数理物理学者のJohn Baezは、こう語りました。
「彼は、カントールやグロタンディックのような天才でした。」

In short, he was a genius akin to Cantor or Grothendieck, at times teetering on the brink of sanity, yet gripped by an immense desire for beauty and clarity, engaging in struggles that gripped his whole soul. From the fires of this volcano, truly original ideas emerge.

重要なことは、彼の仕事は、数学の基礎とコンピュータという二つの世界に関係していました。

今回のセミナーのように、「コンピュータと数学」という問題を取り上げるのなら、Veovodskyは 21世紀の最も重要な人物の一人だと僕は考えています。

彼の数学の分野での仕事、Homotopy Type Theory (HoTT と呼ばれます) と Univalent Foundation については、このセミナーの後のセッションで概略を紹介しようと思います。

このセッションでは、「数学とコンピュータ」について、彼がどういうことを考えていたのか、彼のUniMath プロジェクトを中心に紹介しようと思います。


数学の証明にコンピュータを利用しよう

数学者も間違いを犯すとことがある

数学は厳密な証明によって構成されていると思われています。

でも、意外に思われるかもしれませんが、数学には、「間違った証明」という問題があります。数学者も間違いを犯すことがあるということです。

Veovodskyにも、苦い経験がありました。

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きます。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 彼の「フィールズ賞受賞」という「権威」によって、その証明を「信じて」いました。

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年になってからでした。

別のこともありました。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のです。結論的には、彼は、正しかったのですが、彼が、最終的に、自分が正しいことを確信できたのは、2013年になってからでした。

Voevodskyは、考えます。「数学が、累積的（accumulative）な性格を持つのなら、もしも、そこに誤りが紛れ込むと、それも、累積する可能性がある。」と。

数学の証明にコンピュータを利用しよう

こうした問題に対する、Veovodskyのとった対応はラジカルで驚くべきものでした。

彼は、数学をその基礎から改めて考えなおしたのです。そして、集合論に変わる新しい数学の基礎づけ Univalent Foundationを構築します。

さらに、彼は、数学の証明に、コンピュータを使うべきだと主張した最初の数学者となりました。それは革命的な主張です。

そして、そのためのコンピュータによる証明支援システムのライブラリーUniMathを開発しました。

彼の考えたことを、少し長くなりますが、彼のインタビュー記事から引用します。

彼は、「数学は今、危機的状況、いや、2つの危機に瀕している」といいます。

「第一は、それは、「純粋数学」と「応用数学」の分離に関連しています。

遅かれ早かれ、実用性のないことに従事している人たちに、なぜ社会がお金を払わなければならないのかという疑問が生じるのは明らかでした。」

「第二は、あまり明らかではありませんが、純粋数学の複雑さに関連しています。

遅かれ早かれ、論文が複雑すぎて詳細な検証ができなくなり、発見されないエラーが蓄積される過程が始まるということです。

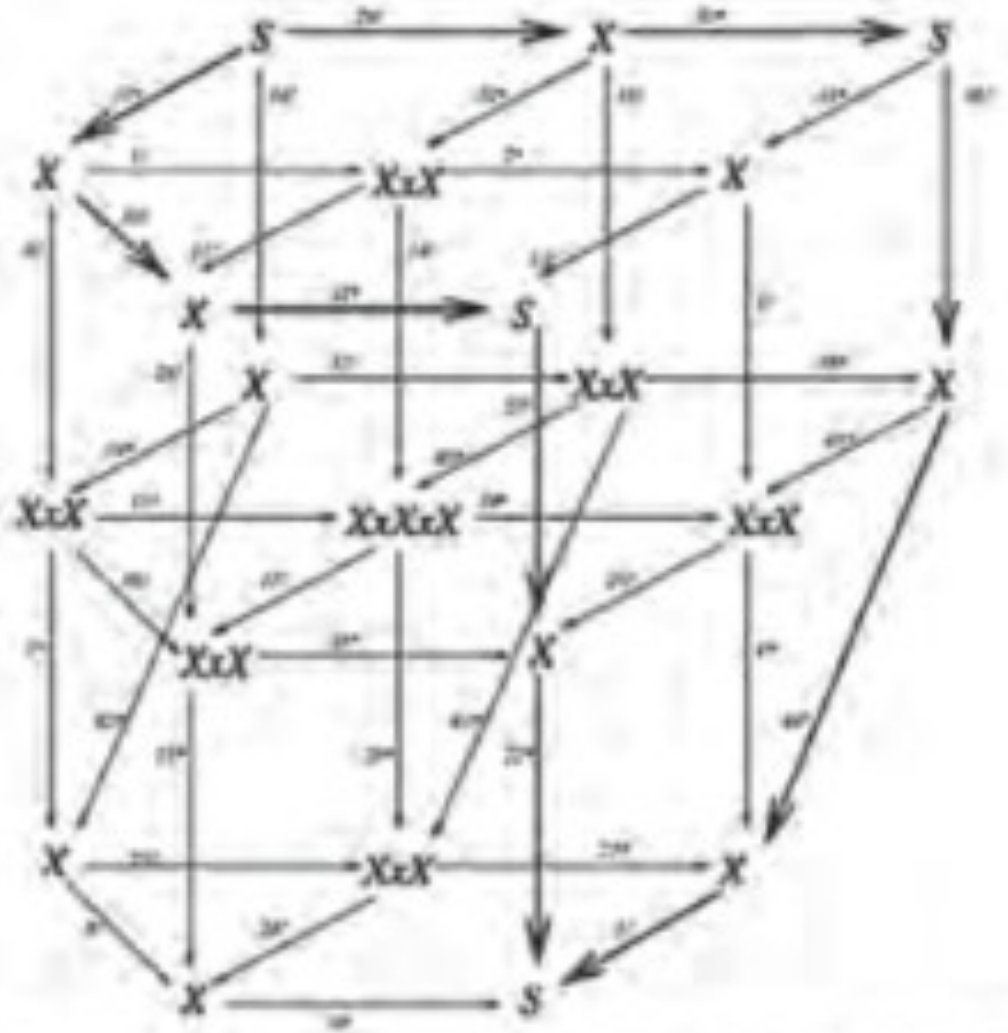
数学は非常に奥の深い学問であり、ある論文の結果が、通常、何本も何本も前の論文の結果に依存するという意味で、数学にとってのこの誤りの蓄積は非常に危険です。

そしてすぐに、唯一の長期的な解決策は、私が抽象的、論理的、数学的に構築したものを、コンピューターを使って検証できるようにすることだということが明らかになりました。」

ただ、すぐにコンピュータが使えたわけではありません。当時、そうしたツールは存在しなかったし、何よりも、そうした手段を使うには、数学の基礎づけ自体が、不十分なものでした。

「既存の基礎体系には2つの大きな問題があり、それらは不十分なものでした。

第一に、既存の数学の基礎は、述語論理の言語に基づいており、このクラスの言語はあまりにも限定的でした。第二に、既存の基礎は、例えば、私の2-theoryに関する研究のような対象に関する記述を直接表現するために使用することができませんでした。」



For the convenience of further reference we numbered all the arrows. The right vertical face of the diagram is the diagram (2) defining the 2-morphism $Id \to \Omega\Sigma$ and the upper horizontal face is the diagram (1) defining the 2-morphism $\Sigma\Omega \to Id$. The whole diagram is the union of the front part which

Library UniMath.Foundations.Preamble

Introduction. Vladimir Voevodsky . Feb. 2010 - Sep. 2011

This is the first in the group of files which contain the (current state of) the mathematical library for the proof assistant Coq based on the Univalent Foundations. It contains some new notations for constructions defined in Coq.Init library as well as the definition of dependent sum.

Initial setup unrelated to Univalent Foundations

```
Require Export UniMath.Foundations.Init.
```

Universe structure

UniMath プロジェクト

```
Definition UU := Type.

Identity Coercion fromUUtoType : UU >-> Sortclass.
```

UniMath プロジェクト

こうして彼は、彼が発見したUnivalence Axiomを中心とするHomotopy Type Theoryを武器に、集合論・カテゴリー論論に代わるUnivalent Foundationという数学の新しい基礎付けを開始することになります。

興味深いのは、彼が、こうした理論展開を、数学論文ではなく、Coqのプログラムの形でGitHubで公開したことでした。

https://github.com/vladimirias/Foundations/

彼は、全ての数学の領域を、このUnivalent Foundationの上で、コンピュータで検証可能なプログラムとして記述して基礎づけようとします。

この壮大なプロジェクトがUniMathです。20世紀に、ブルバキがやったことを、本ではなくコンピュータのプログラムとして展開しようとしたのです。

UniMathについて、彼は次のように述べています。

「UniMathは構成的数学のライブラリですが、アルゴリズムに関するものではありません。ユーザーは、UniMathをコンパイルされたコードの形で見ることはありません。

その価値は、さまざまなユーザー入力に対応して、さまざまなアウトプットを提供する能力にあります。

UniMathは、ソースコードの形で存在します。UniMathを「使う」ことは、既存のソースコードを拡張して新しいソースコードを書くことです。

それを行うためにには、以前に書かれたソースを読み、それをを理解しなければなりません。

人によって、UniMathを使う目的は様々でしょう。

ある人は、複雑な数学の証明を検証したいかもしれません。

ある人は、数学のある古典的領域の問題を、構成主義的数学に枝分かれさせるのに使いたいと思うかもしれません。

ある人は、学生に厳密な数学とは何かを教えるための教材として使うかもしれません。」

以前、数学的発見が、コンピュータの世界におちてくるまで、40年近い「タイムラグ」があるという話をしました。

ただ、今回のVeovodskyによる数学の新しい基礎の発見には、こうした「タイムラグ」はありません。発見した数学者自身が、コンピュータのライブラリーを開発し、公開しているのですから。数学とコンピュータの距離は、大きく近づきました。

きわめて残念なことに、Veovodskyは、とつぜんなくなります。

UniMathの開発は、Veovodskyの死によっ、一旦中断するのですが、後継者たちによってその開発は継続されています。

広がるビジョン

重要なことは、彼のUniMathのビジョンは、大きく広がろうとしていることです。

coqではなくlean上で開発されているのmathlib という数学の証明支援のライブラリーがあります。彼のビジョンは、mathlibのコミュニティによっても受け継がれていると思います。

“The Lean Mathematical Library”
https://arxiv.org/pdf/1910.09336.pdf

https://github.com/leanprover-community/mathlib

コンピュータに数学の問題を解かせようとした OpenAIのプロジェクト"Theorem Prover"の最大の「学習データ」は、実はこの mathlib のソースコードそのものでした。

OpenAIの意欲的なプロジェクト"Theorem Prover"とその失敗については、2022年3月のマルレクの資料を参照ください。

「コンピュータ、数学の問題を解き始める」
https://www.marulabo.net/docs/math-proof/

今年の7月には、Univalent Mathematics と UniMath libraryを学ぶスクールが開催されていました。

嬉しいニュースです。

School on Univalent Mathematics

July 29 - August 02, 2024, Minneapolis, USA

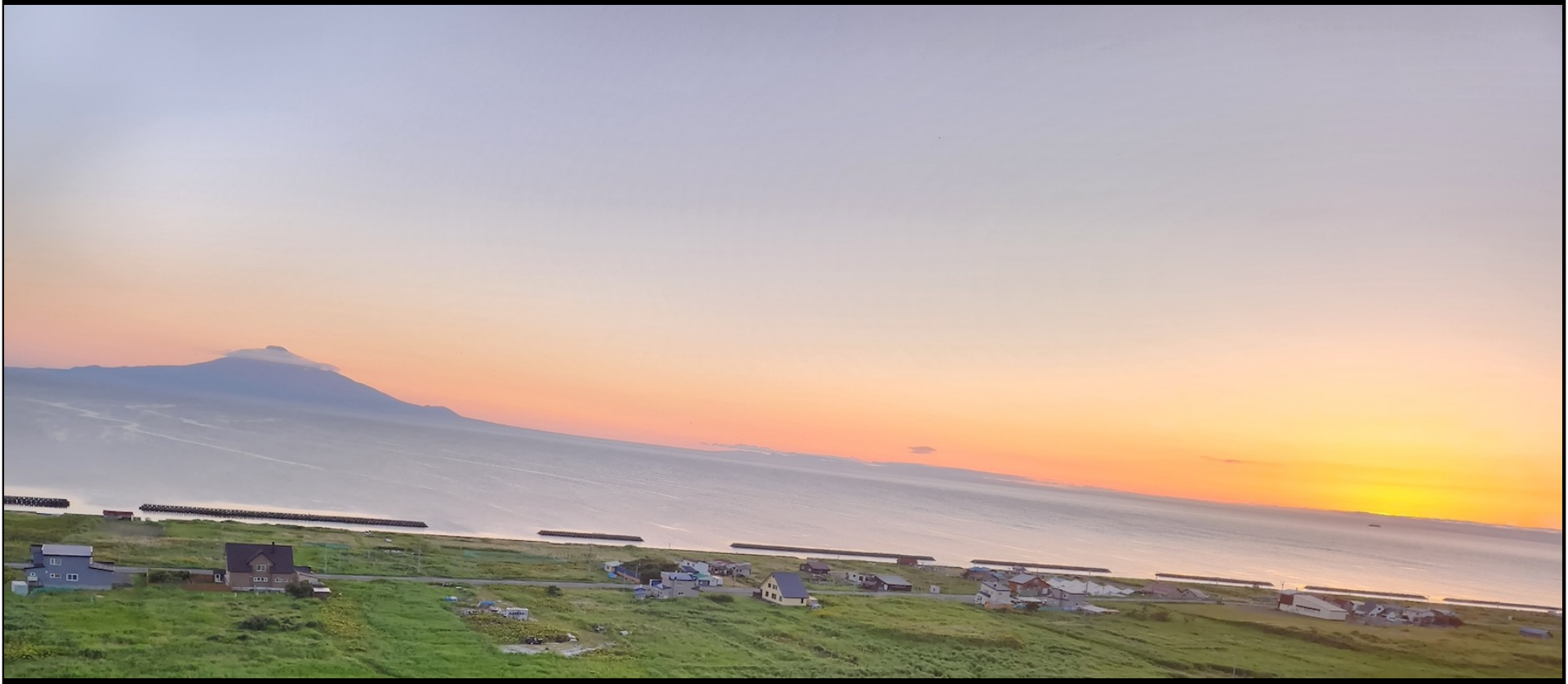
School on Univalent Mathematics

July 29 - August 02, 2024, Minneapolis, USA

Homotopy Type Theory and Univalent Mathematics are emerging fields of mathematics that study a fruitful relationship between homotopy theory and (dependent) type theory. This relation plays a crucial role in Voevodsky's program of Univalent Foundations, a new approach to foundations of mathematics, based on ideas from homotopy theory, such as the Univalence Principle.

https://unimath.github.io/minneapolis2024/

The UniMath library is a large repository of computer-checked mathematics, developed from the univalent viewpoint. It is freely available for everyone, as an open-source project, from the web. The School will give many young researchers an opportunity to familiarize themselves with the UniMath library and become contributors.



Part 2

「計算 ＝ 証明 ＝ プログラム」という認識の発展

Part 2 Agenda
「計算 = 証明 = プログラム」という認識の発展

Curry-Howard対応

Dependent Type Theory -- Martin-Löf

Homotopy Type Theory I -- Dependent Type

Homotopy Type Theory II-- Univalent foundations

Curry-Howard対応

Curry-Howard対応

- 1940年代のChurchの仕事から、ラムダ計算に対する関心が、ふたたび活発化するのは、20年近くたった1960年代からである。

Curry-Howard対応

- 1940年代のChurchの仕事から、ラムダ計算に対する関心が、ふたたび活発化するのは、20年近くたった1960年代からである。
- 理由ははっきりしている。コンピュータが現実に動き出したからである。

Curry-Howard対応

- 1940年代のChurchの仕事から、ラムダ計算に対する関心が、ふたたび活発化するのは、20年近くたった1960年代からである。
- 理由ははっきりしている。コンピュータが現実に動き出したからである。
- この時期の代表的な成果は、HowardのCurry-Howard対応の研究である。

Curryの発見

型付きラムダ計算と
直観主義論理との対応

Curryの発見
型付きラムダ計算と直観主義論理との対応

- 既に1934年に、Curryは、型付きラムダ計算と直観主義論理とのあいだに、対応関係があることを発見していた。

Curryの発見
型付きラムダ計算と直観主義論理との対応

- 既に1934年に、Curryは、型付きラムダ計算と直観主義論理とのあいだに、対応関係があることを発見していた。

- 型を持つラムダ計算の関数の型を示す矢印 → と、論理式“A → B” の中に出てくる含意を意味する矢印 → との間に、対応関係があるというのだ。

Curryの発見
型付きラムダ計算と直観主義論理との対応

- 既に1934年に、Curryは、型付きラムダ計算と直観主義論理とのあいだに、対応関係があることを発見していた。

- 型を持つラムダ計算の関数の型を示す矢印 → と、論理式“A → B” の中に出てくる含意を意味する矢印 → との間に、対応関係があるというのだ。

Curry, Haskell (1934), "Functionality in Combinatory Logic”
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076489/pdf/pnas01751-0022.pdf

Curryの発見
型付きラムダ式の型の定義と論理式

型付きλ式の型の定義。

1. 単純な変数 v_i は型を持つ。 $v_i : \tau$
2. $x : \sigma$ で $e : \tau$ なら、$(\lambda x_\sigma.e) : (\sigma \rightarrow \tau)$
3. $e_1 : (\sigma \rightarrow \tau)$で、$e_2 : \sigma$ なら、$(e_1\ e_2) : \tau$

Curryの発見
型付きラムダ式の型の定義と論理式

型付きλ式の型の定義。

1. 単純な変数 v_i は型を持つ。 $v_i : \tau$
2. $x : \sigma$ で $e : \tau$ なら、$(\lambda x_\sigma.e) : (\sigma \rightarrow \tau)$
3. $e_1 : (\sigma \rightarrow \tau)$で、$e_2 : \sigma$ なら、$(e_1\ e_2) : \tau$

Curryの発見
型付きラムダ式の型の定義と論理式

型付きλ式の型の定義。

1. 単純な変数 v_i は型を持つ。 $v_i : \tau$
2. $x : \sigma$ で $e : \tau$ なら、$(\lambda x_\sigma.e) : (\sigma \rightarrow \tau)$
3. $e_1 : (\sigma \rightarrow \tau)$で、$e_2 : \sigma$ なら、$(e_1\ e_2) : \tau$

この型付きラムダ計算は、

「A → B で A なら Bである」（三段論法）

という論理式と同じ構造をしている。

Curryの発見
型付きラムダ式の型の定義と論理式

型付きλ式の型の定義。

1. 単純な変数 v_i は型を持つ。 $v_i : \tau$
2. $x : \sigma$ で $e : \tau$ なら、$(\lambda x_\sigma.e) : (\sigma \rightarrow \tau)$
3. $e_1 : (\sigma \rightarrow \tau)$で、$e_2 : \sigma$ なら、$(e_1\ e_2) : \tau$

この型付きラムダ計算は、
「A → B で A なら Bである」（三段論法）
という論理式と同じ構造をしている。

型を持つラムダ計算の関数の型を示す矢印 → と、
論理式“A → B” の中に出てくる含意を意味する矢印 → との間に、対応関係がある。

Kolmogorovの「証明」の解釈

Kolmogorov
a → b (aならばb)の証明の解釈

- Kolmogorovは、命題 a → b (aならばb)の証明に、「命題 aの証明を命題 bの証明に変換する方法を構築すること」という解釈を与えた。
- このことは、a → bの証明を、aの証明からbの証明への関数と見ることが出来るということを意味する。
- 同様に、次に見るように、命題の意味を、その証明と関連づけることができる。
- Kolmogorovのこの考え（構成主義という）は、Martin-Löfの型の理論に深い影響を与えた。

Kolmogorovの解釈の下では、命題の証明は、何から構成されるか？

□ ⊥（偽）	証明なし
□ **A ∧ B**	Aの証明とBの証明の両方
□ **A ∨ B**	Aの証明、あるいは、Bの証明
□ **A → B**	Aの証明からBの証明を導く方法
□ **(∀x)B(x)**	任意のaに対してB(a)の証明を与える方法
□ **(∃x)B(x)**	あるaに対するB(a)の証明

Kolmogorovの解釈の下では、命題の証明は、何から構成されるか？形式的に

⊥（偽）	none
A ∧ B	Aの証明であるaと、 Bの証明であるbのペア**(a,b)**
A ∨ B	Aの証明である**i(a)**、あるいは、 Bの証明である**j(b)**
A → B	Aの証明であるaに対して、 Bの証明b(a)を与える**(λx)b(x)**
(∀x)B(x)	任意のaに対して Bの証明b(a)を与える**(λx)b(x)**
(∃x)B(x)	あるaと、B(a)の証明であるbのペア **(a,b)**

Howard

論理式はラムダ計算の型と
見なすことが出来る

Howard
論理式はラムダ計算の型と見なすことが出来る

- Howardは、このCurryの発見を、さらに深く考えた。

Howard
論理式はラムダ計算の型と見なすことが出来る

- Howardは、このCurryの発見を、さらに深く考えた。
- 1969年の彼の論文のタイトルが示すように、
  論理式は型付きラムダ計算の型と見なすことが出来るのである。

Howard,Williams (1980)
"The formulae-as-types notion of construction”
http://www.cs.cmu.edu/~crary/819-f09/Howard80.pdf

Howard
論理式はラムダ計算の型と見なすことが出来る

- Howardは、このCurryの発見を、さらに深く考えた。
- 1969年の彼の論文のタイトルが示すように、論理式は型付きラムダ計算の型と見なすことが出来るのである。
- ただし、彼のこの論文が公開されたのは、1980年になってからのことらしい。

Howard,Williams (1980)
"The formulae-as-types notion of construction”
http://www.cs.cmu.edu/~crary/819-f09/Howard80.pdf

Martin-Löf
命題への型の拡張

- Churchの単純な型を持つラムダ計算の体系は、
  基本的には、型A, 型Bに対して、型 A → Bで表現される
  関数型の型しか持たない。

Martin-Löf
命題への型の拡張

- Churchの単純な型を持つラムダ計算の体系は、
  基本的には、型A, 型Bに対して、型 A → Bで表現される
  関数型の型しか持たない。

- それに対して、Martin-Löfは、論理的な命題にも、自然なスタイルで型を定義した。例えば、Aが型であり、Bが型であれば、A∧BもA→BもA∨Bも型であるというように。もちろん、それぞれは異なる型である。

ある 命題 aが型Aを持つ時、***a : A*** で表す。

Curry-Howard 対応

Curry-Howard 対応
「型」と「命題」、「項」と「証明」との対応

- Curry-Howard対応の、中心的な概念は、「型」と「命題」、「項」と「証明」との対応である。

Curry-Howard 対応
「型」と「命題」、「項」と「証明」との対応

- Curry-Howard対応の、中心的な概念は、「型」と「命題」、「証明」と「項」の対応である。

- 「*p* が命題 *P* の証明である」ことを、***p : P***
  と表すことができる。

Curry-Howard 対応
「型」と「命題」、「項」と「証明」との対応

- Curry-Howard対応の、中心的な概念は、「型」と「命題」、「項」と「証明」との対応である。

- 「*p* が命題 *P* の証明である」ことを、***p : P***
  と表すことができる。
- 「*p* は、型 *P* の項である」ことも、***p : P***
  と表すことができる。

Curry-Howard 対応
「型」と「命題」、「項」と「証明」との対応

- Curry-Howard対応の、中心的な概念は、「型」と「命題」、「項」と「証明」との対応である。

- 「*p* が命題 *P* の証明である」ことを、***p : P***
  と表すことができる。
- 「*p* は、型 *P* の項である」ことも、***p : P***
  と表すことができる。

- 「*p* が命題 *P* の証明である」という判断を表す ***p : P*** は、「*p* は、型 *P* の項である」という判断を表す ***p : P*** と見なすことが出来るし、また、逆の見方も出来るのである。

Curry-Howard対応

「型」と「命題」、「項」と「証明」との対応

「p は、命題 P の証明である」

証明　　命題

$$p : P$$

Curry-Howard対応

「型」と「命題」、「項」と「証明」との対応

$$p : P$$

項　　型

「p は、型 P の項である」

Curry-Howard対応

「型」と「命題」、「項」と「証明」との対応

「p は、命題 P の証明である」

証明　　命題

$$p : P$$

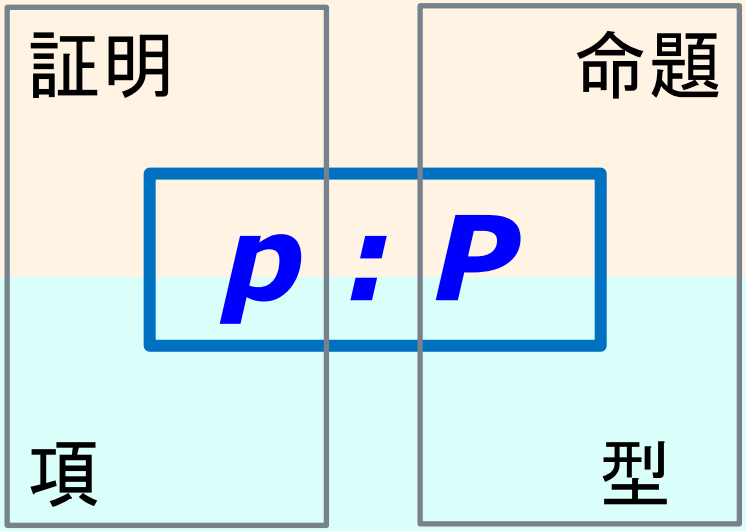
項　　型

「p は、型 P の項である」

Curry-Howard対応

「型」と「命題」、「項」と「証明」との対応

「p は、命題 P の証明である」

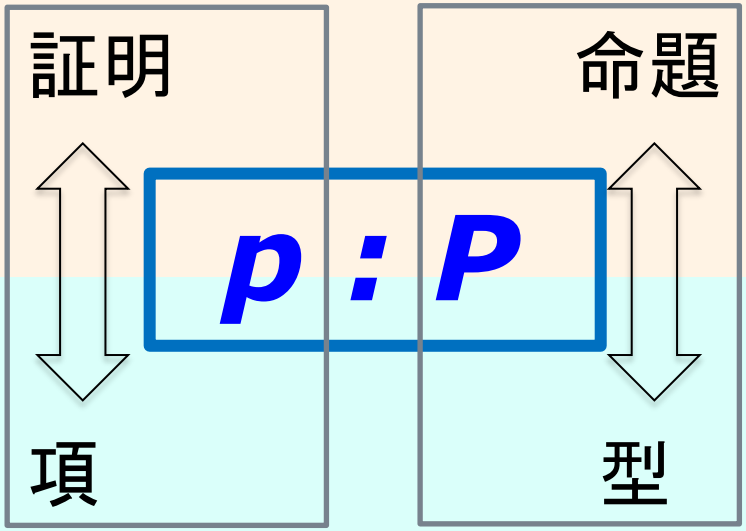


「p は、型 P の項である」

Curry-Howard対応

「型」と「命題」、「項」と「証明」との対応

「p は、命題 P の証明である」

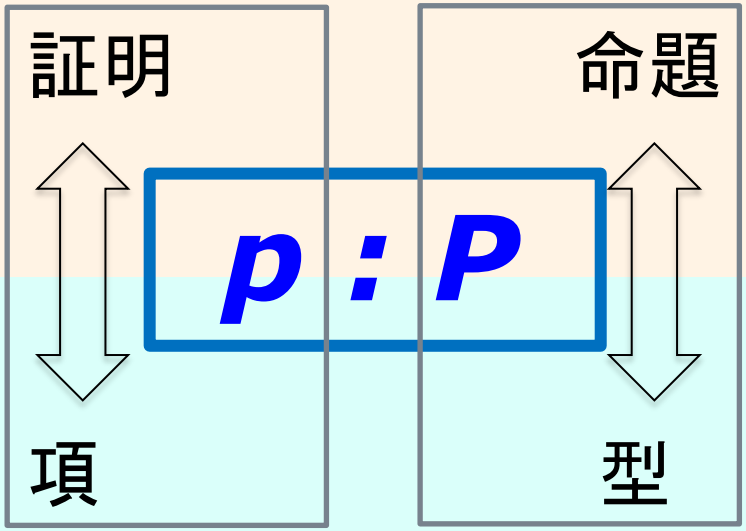


「p は、型 P の項である」

Curry-Howard対応

「型」と「命題」、「項」と「証明」との対応

「p は、命題 P の証明である」



証明は
項である

命題は
型である

「p は、型 P の項である」

“Propositions as Types”
“Proofs as Terms”

“Propositions as Types”
“Proofs as Terms”

- Howardの洞察は、”Propositions as Types”, “Proofs as Terms”（これを、PATという）として、あとにみるMartin-Löfの型の理論に大きな影響を与えた。

“Propositions as Types”
“Proofs as Terms”

- Howardの洞察は、”Propositions as Types”, “Proofs as Terms” (これを、PATという)として、あとにみるMartin-Löfの型の理論に大きな影響を与えた。
- 同時に、Curry-Howard対応は、型付きラムダ計算をプログラムと見なせば、”Proof as Program”としても解釈出来る。

“Propositions as Types”
“Proofs as Terms”

- Howardの洞察は、”Propositions as Types”, “Proofs as Terms” (これを、PATという)として、あとにみるMartin-Löfの型の理論に大きな影響を与えた。
- 同時に、Curry-Howard対応は、型付きラムダ計算をプログラムと見なせば、”Proof as Program”としても解釈出来る。
- こうした観点は、今日のCoq等の証明支援言語の理論的基礎になっている。

$a : A$ の解釈

$a : A$ の解釈

- 論理＝数学的には、**a : A** には、次のような様々な解釈がある。

1. 集合論: Russellの立場
   Aは集合であり、aはその要素である。$a \in A$
2. 構成主義: Kolmogorovの立場
   Aは問題であり、aはその解決である
3. PAT: Curry & Howardの立場
   Aは命題であり、aはその証明である。 $a : A$
4. 型の理論: Martin-Löf
   Aは型であり、aはその項である。 $a : A$
5. HoTT: Voevodskyの立場
   Aは空間であり、aはその点である。 $a : A$

$a : A$ の解釈

- 論理＝数学的には、**a : A** には、次のような様々な解釈がある。

1. 集合論: Russellの立場
   Aは集合であり、aはその要素である。$a \in A$
2. 構成主義: Kolmogorovの立場
   Aは問題であり、aはその解決である
3. PAT: Curry & Howardの立場
   Aは命題であり、aはその証明である。 $a : A$
4. 型の理論: Martin-Löf
   Aは型であり、aはその項である。 $a : A$
5. HoTT: Voevodskyの立場
   Aは空間であり、aはその点である。 $a : A$

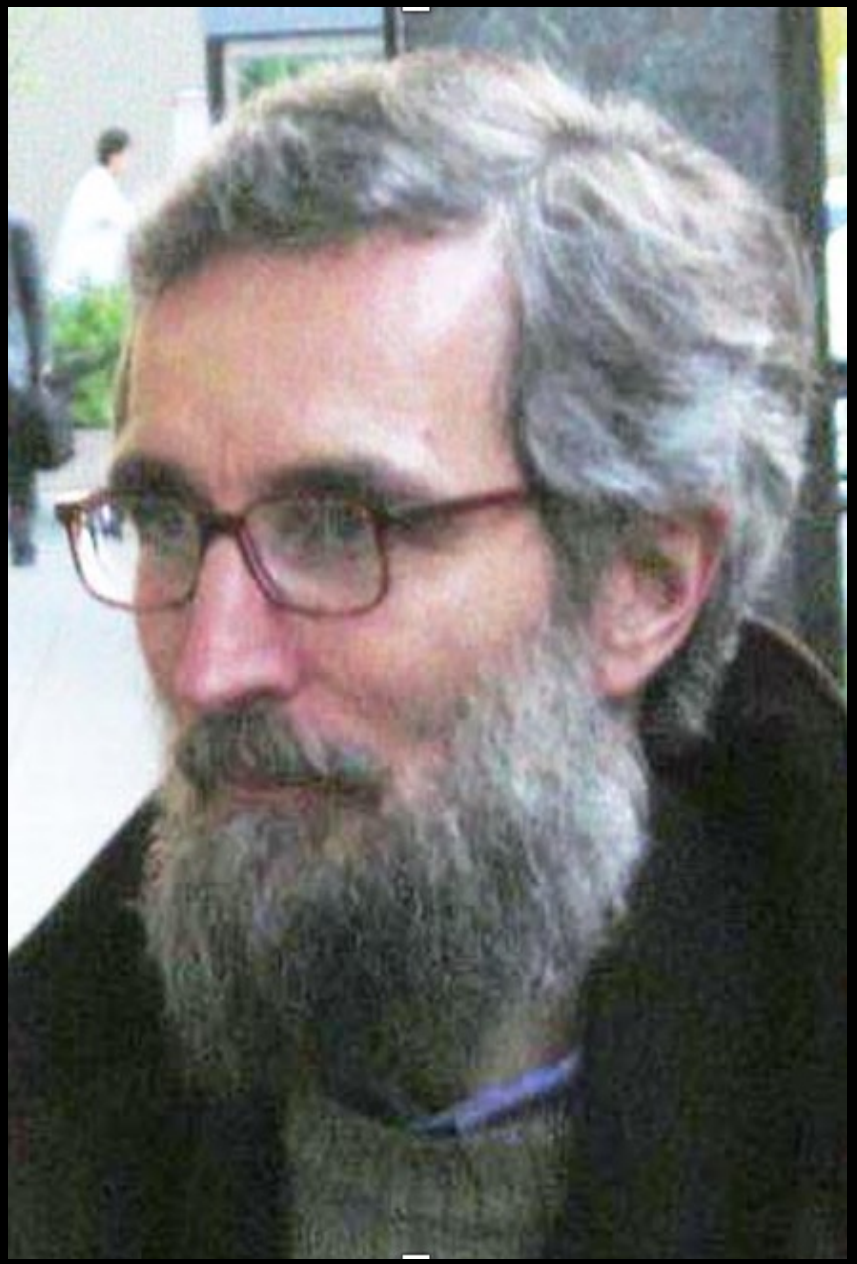
a : A の解釈

□ 論理＝数学的には、**a : A** には、次のような様々な解釈がある。

1. 集合論: Russellの立場
   Aは集合であり、aはその要素である。 $a \in A$
2. 構成主義: Kolmogorovの立場
   Aは問題であり、aはその解決である
3. PAT: Curry & Howardの立場
   Aは命題であり、aはその証明である。 $a : A$
4. 型の理論: Martin-Löf
   Aは型であり、aはその項である。 $a : A$
5. HoTT: Voevodskyの立場
   Aは空間であり、aはその点である。 $a : A$

Curry-Howard 対応

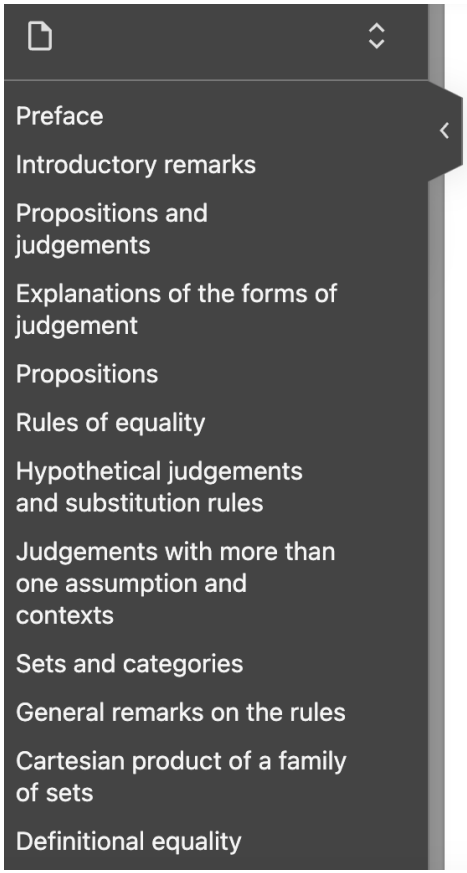
Per Martin Löf
Dependent Type Theory



Dependent Type Theory

現在の「型の理論」の基本が出来上がるのは、1980年代になってからです。その中心人物は、Martin-Löfです。

現在のCoq, Agda, leanという証明支援システムは、彼のDependent Typeの理論に基礎付けられています。



Intuitionistic Type Theory

Per Martin-Löf

Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980

https://archive-pml.github.io/martin-lof/pdfs/Bibliopolis-Book-retypeset-1984.pdf

命題への型の拡張

Churchの単純な型を持つラムダ計算の体系は、
基本的には、型A, 型Bに対して、型 A → Bで表現される関数型の型しか持ちませんでした。

それに対して、Martin-Löfは、論理的な命題にも、自然なスタイルで型を定義しました。
例えば、Aが型であり、Bが型であれば、A∧BもA→BもA∨Bも型であるというように。もちろん、それぞれは異なる型です。

ある aが型Aを持つ時、***a : A*** で表します。

同一性への型の付与

Martin-Löf は、式 a = b にも型を与えます。

$$a =_A b : Id(A\ a\ b)$$

型 Id(A a b) を持つ式は、a と b は等しいという意味を持ちます。$a =_A b$ のAは、型Aの中での同一性であることを表しています。すなわち、a : A で b : Aで、かつa = bの時にはじめて、$a =_A b$ は型Id(A a b)を持ちます。

ここでは式(＝項)の同一性について述べたのですが、項の同一性と型の同一性は、異なるものです。

型の理論の記述

Martin-Löfの型の理論は、次の四つの形式で記述されます。

1. ある対象aが、型であること
   $a : Type$
2. ある表現式 aが、型 αの項であること
   $a : \alpha$
3. 二つの表現式が、同じ型αの内部で等しいこと
   $a =_\alpha b : \alpha$
4. 二つの型が、等しいこと
   $\alpha = \beta \ : Type$

Dependent Type (従属型)

これまで、元になる型 A, B を指定すると A∧B, A → B, A∨B といった、新しい型が決まるといったスタイルで型を導入してきました。

これとは異なる次のようなスタイル、型Aそのものではなく型Aに属する要素 a : Aに応じて新しい型 B(a)が変化するような型の導入が可能です。

例えば、実数R上のn次元のベクトル Vec(R,n)とn+1次元のベクトルVec(R,n+1)は、別の型を持つのですが、これはnに応じて型が変わると考えることが出来ます。

こうした型をDependent Typeと呼び、次のように表します。

$$\Pi(x:A).B(x)$$

Dependent Typeは、ある型Aの値aにディペンドして変化する型です。

先の例のベクトルの型は、次のように表されます。

$$\Pi(x:N).Vec(R,x)$$

これは、全てのnについてVec(R,n)を考えることに対応しています。型の理論では、全称記号は、Dependent Typeとして導入されます。

Dependent TypeとPolymorphism

Dependent Typeの例を、もう一つあげましょう。

n次元のベクトルは、自然数 N、整数 Z、実数 R、複素数 C 上でも定義出来ます。{ N, Z, R, C }：T とする時、次のように定義されるDependent Typeを考えましょう。

$$\Pi(x : T).Vec(x,n)$$

これは、次元nは同じだが、定義された領域が異なる別の型Vec(N,n),Vec(Z,n),Vec(R,n), Vec(C,n)を考えることに相当します。

こうして、Polymorphicな関数は、Dependent Typeとして表現されることが分かります。

Inductive Type

型の理論では、基本的な定数と関数から、新しい型を帰納的に定義することが出来ます。こうした型をInductive Type（帰納的型）と呼びます。

次は、そうした帰納的な定義の例です。
ここでは、自然数の型natが、ゼロを意味する定数Oとsuccessor関数を表す関数Sで、帰納的に定義されています。

```
Inductive nat : Type :=
  | O : nat
  | S : nat -> nat.
```

論理とは何か？
Martin-Löfが考えたこと

Martin-Löf の研究スタイルには、普通の数学者にはない特徴があると、僕は考えています。

それは、「論理とは何か？」について、いわば「哲学的」に、深く考える姿勢を持っているところにあると思います。

ここは、そうした彼の考察を紹介しようと思います。

Per Martin-Löf

On the Meanings of the Logical Constants and the Justifications of the Logical Laws

Preface

The following three lectures were given in the form of a short course at the meeting Teoria della Dimostrazione e Filosofia della Logica, organized in Siena, 6–9 April 1983, by the Scuola di Specializzazione in Logica Matematica of the Università degli Studi di Siena. I am very grateful to Giovanni Sambin and Aldo Ursini of that school, not only for recording the lectures on tape, but, above all, for transcribing the tapes produced by the recorder: no machine could have done that work.

https://ncatlab.org/nlab/files/MartinLofOnTheMeaning96.pdf

論理式の意味

論理式の意味というと、次のようなことを思い浮かべるかもしれません。

論理式	論理式の意味
A∧B	AかつB
A∨B	AまたはB
A→B	AならばB
~A	Aではない
∀xP(x)	全てのxについてP(x)が成り立つ
∃xP(x)	あるxが存在してP(x)が成り立つ

「Aは真である」の意味

論理式の意味というと、先のようなことを思い浮かべるかもしれません。

ただ、それは間違いではないのですが、不十分です。

例えば、ある論理式で表される「命題Aが真である」ということの意味を考えてみましょう。

「Aは真である」の意味

「Aは真である」 ということは、

「私は「Aは真である」ことを知っている」ということです。

ただ、その前に、知っていることがあります。

それは、「私は「Aは論理式である」ことを知っている」ということです。

判断(Judgement)と命題

「Aは真である」 あるいは「Aは論理式である」といった言明を「判断」と呼びます。

「判断」の概念は、常に、「命題」の概念に先立ちます。

「判断」における論理的帰結の概念は、「命題」における含意より先に、説明されなければならないのです。

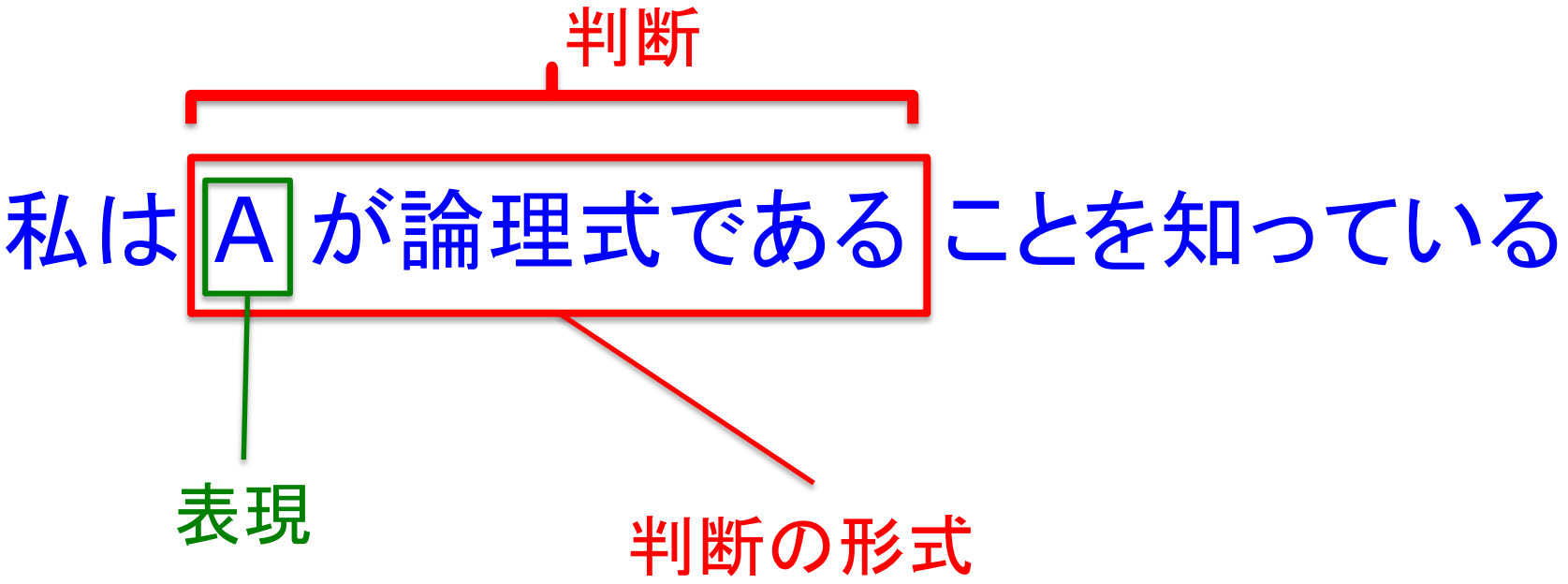
論理式の表現･構成についての判断の構造

私は A が論理式である ことを知っている

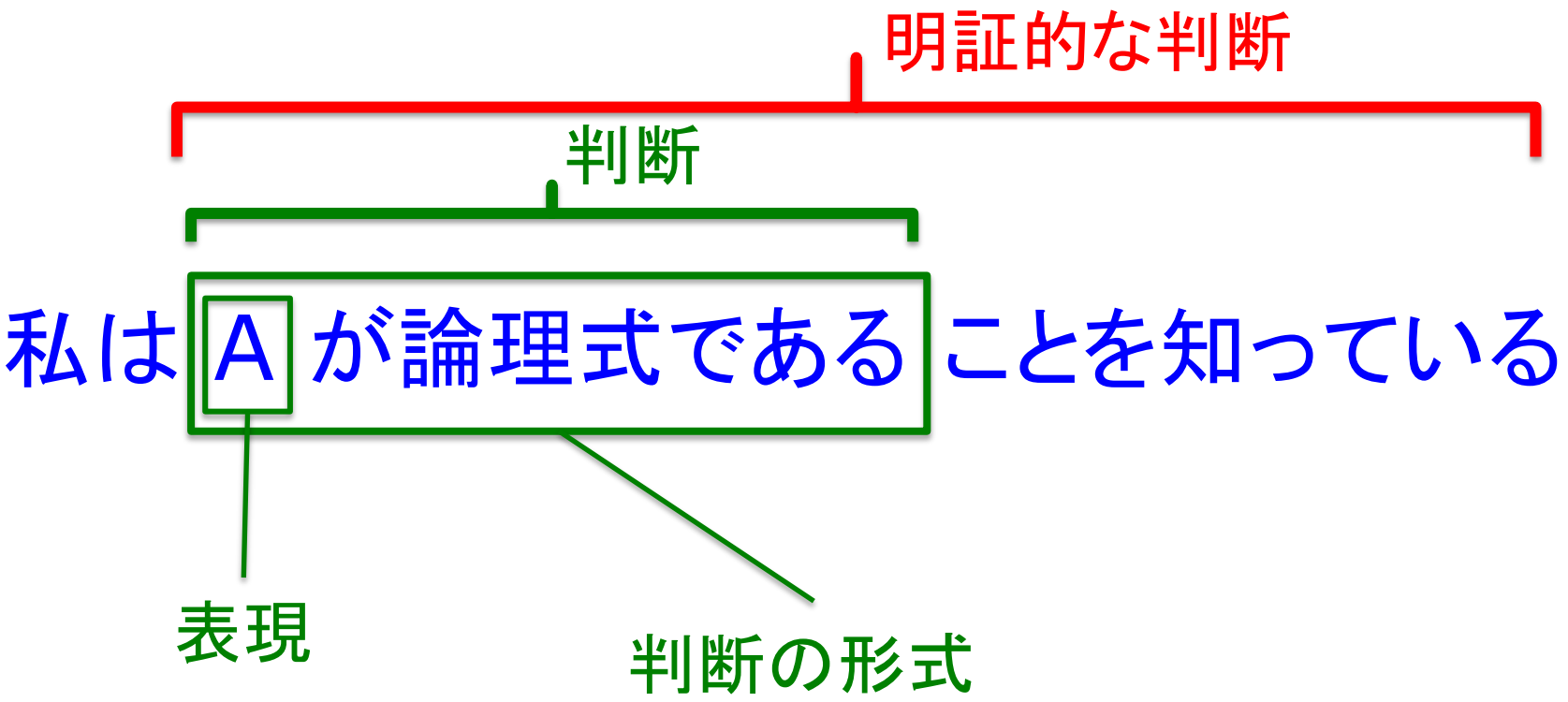
論理式の表現・構成についての判断の構造

私は A が論理式である ことを知っている

論理式の表現・構成についての判断の構造



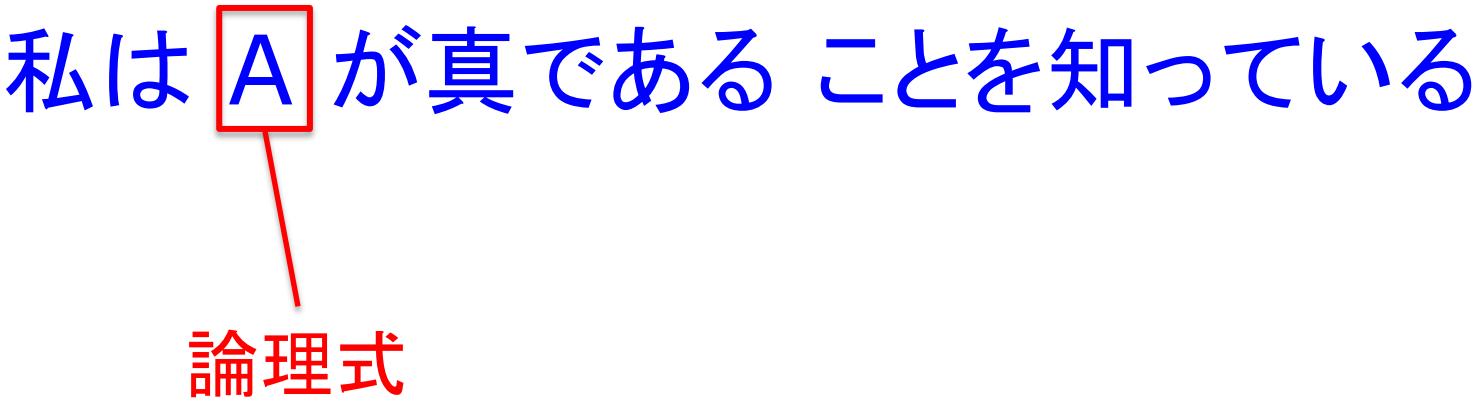
論理式の表現・構成についての判断の構造



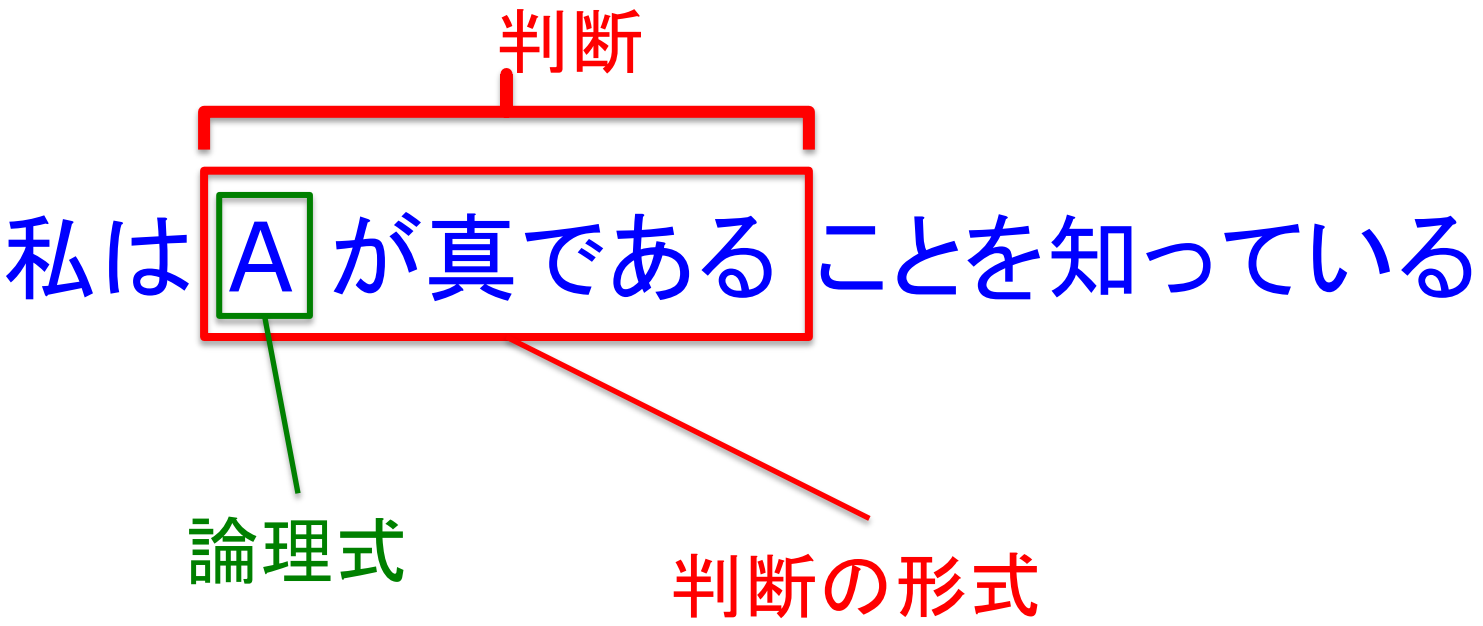
論理式の正しさについての判断の構造

私は A が真である ことを知っている

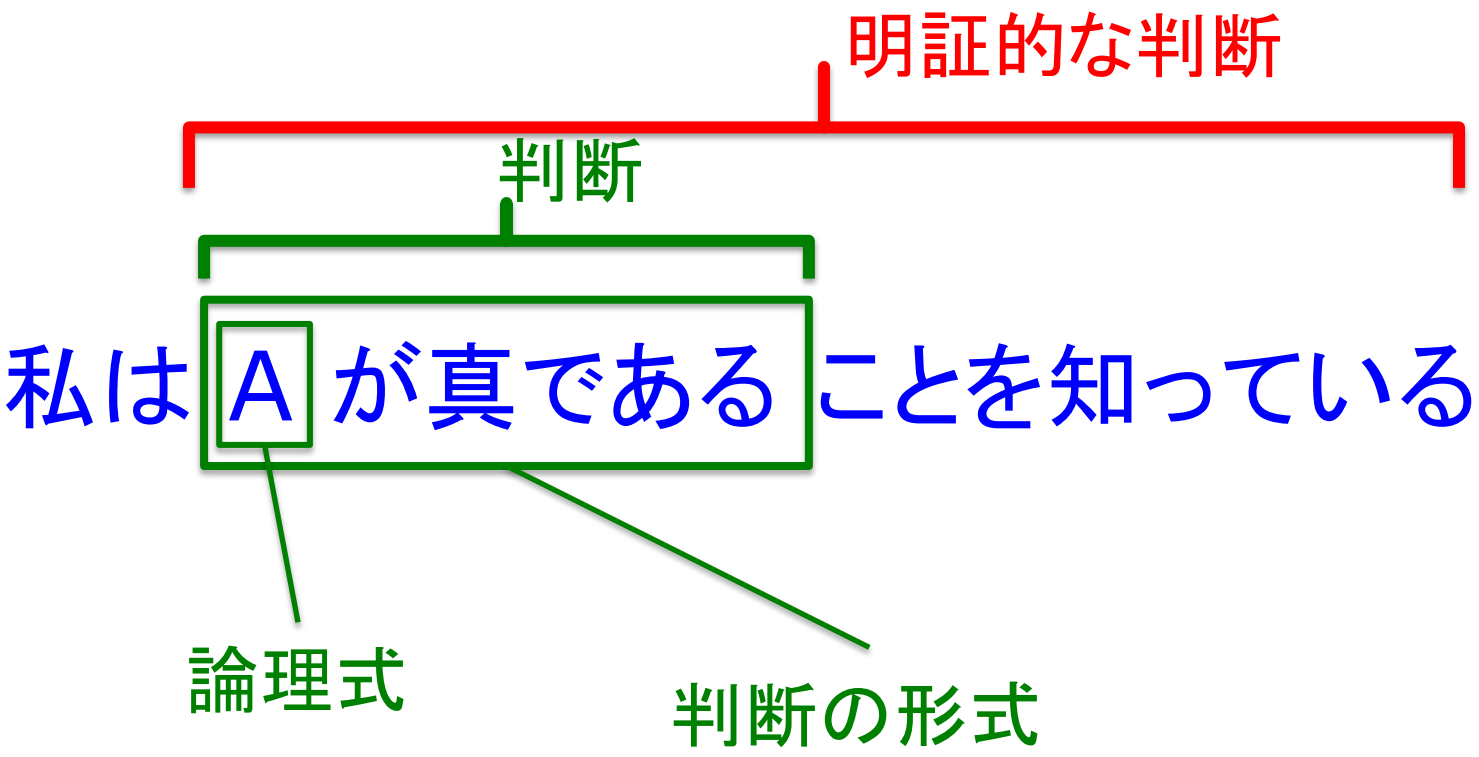
論理式の正しさについての判断の構造



論理式の正しさについての判断の構造



論理式の正しさについての判断の構造



証明とは何か？

証明とは、判断を明証なものにするもの。
証明すること＝知ること＝理解して把握すること
知るようになる＝知識を得ること
証明と知識は、同じもの

判断の論理的帰結の表現

論理学者は、横棒が好きである。

「判断２」が「判断１」の論理的帰結であることを次のように表す。

$$\frac{\text{判断１}}{\text{判断２}}$$

こうした表現を用いて、論理式の構成ルールや、論理式の推論ルールを表現することができます。

判断を表す記号 ‘⊢’
‘turnstile’

「命題Aは真である」という「判断」を、⊢ Aと表すことにしましょう。

「命題Aが真」で、かつ、「命題Bが真」という判断があるなら、「命題 (A ∧ B)は真」であるという判断が成り立ちます。

この推論ルールを次のように表します。

$$\frac{\vdash A \qquad \vdash B}{\vdash (A \wedge B)}$$

もしも、上段、下段に登場しているのが「～は真である」という判断であることが明確であれば、次のように記号⊢ を省略しても構いません。

$$\frac{A \qquad B}{(A \wedge B)}$$

判断の基本的な形式
context ⊢ conclusion

- Γ ctx
  Γは、well-formed なcontextである

$$\frac{}{()\text{ctx}} \qquad \frac{\Gamma\text{ctx} \quad \Gamma \vdash A\text{type}}{(\Gamma, a{:}\,A)\text{ctx}}$$

- Γ ⊢ A type
  Aは、context Γのもとで well-typed な従属型である

- Γ ⊢ a:A
  aは、context Γのもとで型Aのwell-typedな項である

判断は、次の形をしている

context $\vdash$ conclusion

Contexts & judgments

Γ	sequence of variable declarations $(x_1 : A_1), (x_2 : A_2(x_1)), \ldots, (x_n : A_n(\vec{x}_i))$
$\Gamma \vdash A$	A is well–formed **type** in context Γ
$\Gamma \vdash a : A$	**term** a is well-formed and of type A
$\Gamma \vdash A \equiv B$	types A and B are **convertible**
$\Gamma \vdash a \equiv b : A$	a is convertible to b in type A

$$(x : \mathsf{Nat}), (f : \mathsf{Nat} \to \mathsf{Bool}) \vdash f(x) : \mathsf{Bool}$$

等号の定義

- Γ ⊢ A≡A′ type

  AとA′は、context Γのもとで 等しいwell-typedな型であると判断される

- Γ ⊢ a≡a′:A

  aとa′は、context Γのもとで Aの等しいwell-typedな項であると判断される

等号の性質

- 反射性

$$\frac{\Gamma \vdash A\mathrm{type}}{\Gamma \vdash A \equiv A\mathrm{type}}$$

$$\frac{\Gamma \vdash A\mathrm{type} \quad \Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A}$$

- 対称性

$$\frac{\Gamma \vdash A \equiv B\mathrm{type}}{\Gamma \vdash B \equiv A\mathrm{type}}$$

$$\frac{\Gamma \vdash A\mathrm{type} \quad \Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A}$$

等号の性質

- 推移性

$$\frac{\Gamma \vdash A \equiv B\mathrm{type} \quad \Gamma \vdash B \equiv C\mathrm{type}}{\Gamma \vdash A \equiv C\mathrm{type}}$$

$$\frac{\Gamma \vdash A\mathrm{type} \quad \Gamma \vdash a \equiv b : A \quad b \equiv c : A}{\Gamma \vdash a \equiv c : A}$$

等号の性質

- 等しい項の代入の原理

$$\frac{\Gamma \vdash A\mathrm{type} \quad \Gamma \vdash a \equiv b : A \quad \Gamma, x : A, \Delta \vdash B\mathrm{type}}{\Gamma, \Delta[b / x] \vdash B[a / x] \equiv B[b / x]\mathrm{type}}$$

$$\frac{\Gamma \vdash A\mathrm{type} \quad \Gamma \vdash a \equiv b : A \quad \Gamma, x : A, \Delta \vdash c : B}{\Gamma, \Delta[b / x] \vdash c[a / x] \equiv c[b / x] : B[b / x]}$$

- 変数の変換

$$\frac{\Gamma \vdash A \equiv B\mathrm{type} \quad \Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, x : B, \Delta \vdash \mathcal{J}}$$

Homotopy Type Theory I
Dependent Type

新しい型の理論
Homotopy Type Theory

このセッションでは、Voevodsky が中心となって構築した、新しい型の理論である Homotopy Type Theory (HoTT) を紹介しようと思います。

Homotopy Type Theory (HoTT)は、数学の一分野である「ホモトピー論」と、論理学・計算機科学の一分野である「型の理論」とのあいだに、深い結びつきがあるという発見に端を発しています。

ここでいう「型の理論」は、前回のセッションで紹介した Martin-LöfのDependent Type Theory（従属型理論）です。

HoTTでの a:Aの解釈
aは「点」であり、Aは「空間」である

Martin-LöfのDependent Type Theory では、a:A に、「Aは型であり、aはその項である」という解釈を与えます。ここで型Aは、基本的には論理式に対応し、項a はその論理式の証明に対応します。（Curry-Howard対応）

VeovodskyのHomotopy Type Theory では、a:Aは、それとは全く違った解釈が与えられます。そこでは、「Aは「空間」であり、aはその空間上の「点」である」という解釈が与えられます。

どういうことでしょう？

$a : A$ の解釈

論理＝数学的には、**a : A** には、次のような様々な解釈があります。

1. 集合論: Russellの立場
   Aは集合であり、aはその要素である。$a \in A$
2. 構成主義: Kolmogorovの立場
   Aは問題であり、aはその解決である
3. PAT: Curry & Howardの立場
   Aは命題であり、aはその証明である。 $a : A$
4. 型の理論: Martin-Löf
   Aは型であり、aはその項である。 $a : A$
5. HoTT: Voevodskyの立場
   Aは空間であり、aはその点である。 $a : A$

a : A の解釈

論理＝数学的には、**a : A** には、次のような様々な解釈があります。

1. 集合論: Russellの立場
   Aは集合であり、aはその要素である。$a \in A$
2. 構成主義: Kolmogorovの立場
   Aは問題であり、aはその解決である
3. PAT: Curry & Howardの立場
   Aは命題であり、aはその証明である。 $a : A$
4. 型の理論: Martin-Löf
   Aは型であり、aはその項である。 $a : A$
5. HoTT: Voevodskyの立場
   Aは空間であり、aはその点である。 $a : A$

このセッションと次のセッションで、Dependent Type Theoryのの Dependent Type (従属型)の導入と「同一性」への注目という二つの特徴が Homotopy Type Theory ではどのように扱われているかをみていこうと思います。

このセッションでは、Homotopy Type TheoryでのDependent Typeの扱いを見ていきます。

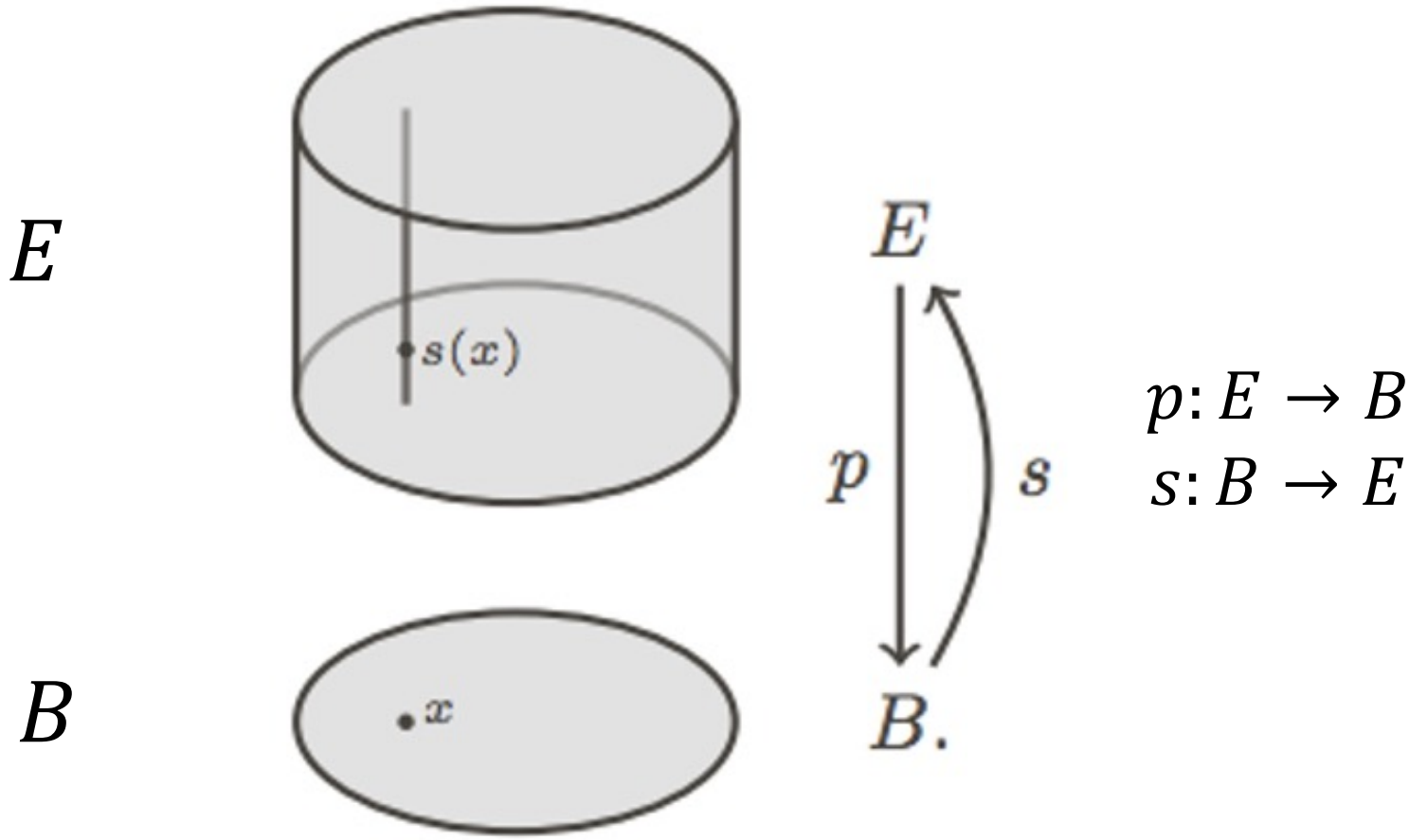
次のセッションで、Homotopy Type TheoryでのUnivalent Foundation という同一性の扱いを見ていこうと思います。

Homotopy Type Theoryでの Dependent Typeの解釈

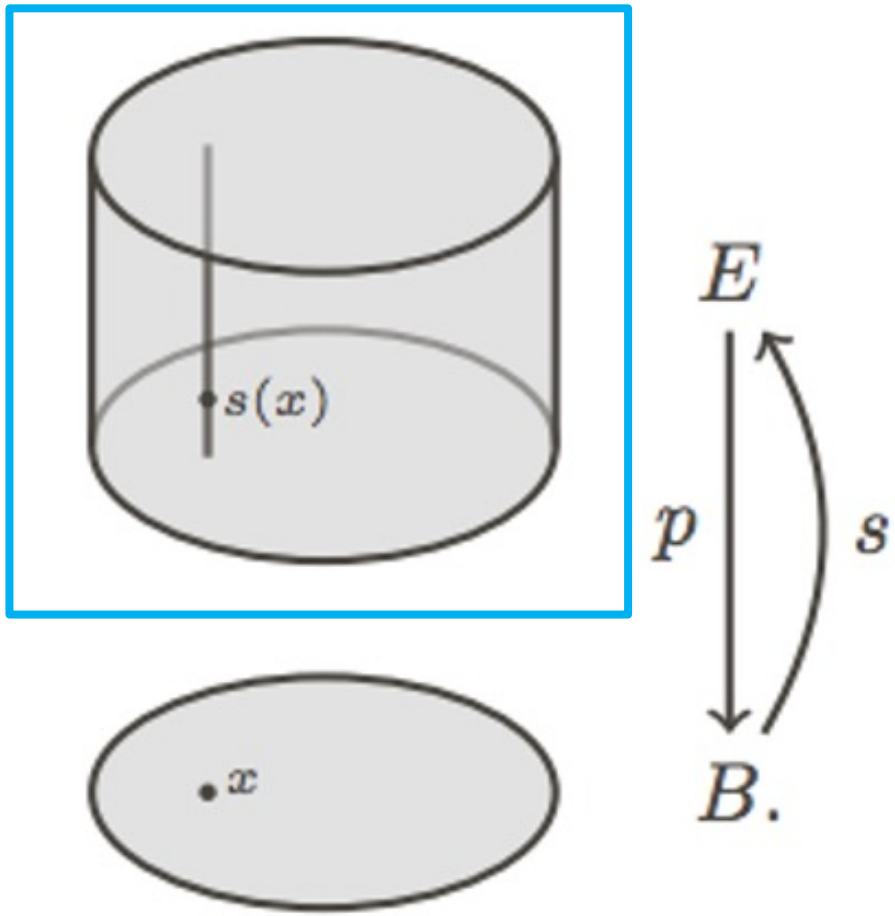
Dependent Type Theoryでは、型Bに属する要素 b： Bに応じて型 E(b)が変化するような型E(b)の導入が可能です。こうした型をDependent Typeと呼び、それが型であることを次のように表します。

$$\Pi(x{:}\,B).\,E(x)\ :\ Type$$

Homotopy Type Theory でも、この$\Pi(x{:}\,B).\,E(x)$ を、Bの値でパラメーター化された$(E_x)_{x\in B}$ と考えるのですが、EとBの関係を次の図のように考えます。


E
s(x)
B
x
E
p
s
B.
p: E → B
s: B → E

Dependent Type
$\Pi(x{:}B).E(x)$ は
この「空間」内の
fiberである



Grothendieck construction

こうした、ベースになる空間Bから、新しい空間Eを構成することを、数学的には、Grothendieck construction と呼びます。

$p: E \to B$をB上のfibrationと呼びます。

Homotopy Type Theory では、Dependent Type $\Pi(x: B).E(x)$ は、B上のfibrationとして表現されます。

Dependent Typeの項の表現

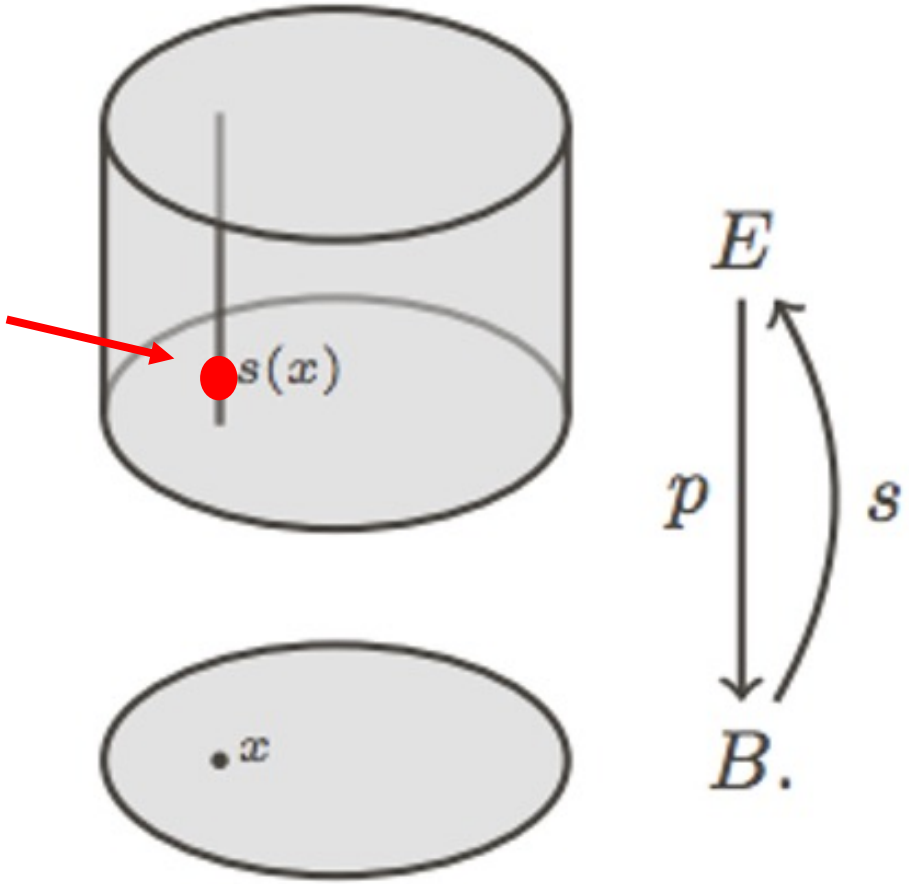
Martin-LöfのDependent Type Theory では、
ある対象aが、型であることは $a : Type$ と表現されますが、
ある表現式 aが、型 αの項であることは $a : \alpha$ と、別の表現を持ちます。

Homotopy Type Theory では、Dependent Typeの項は、どのように表現されるのでしょう。

次の図を見てください。

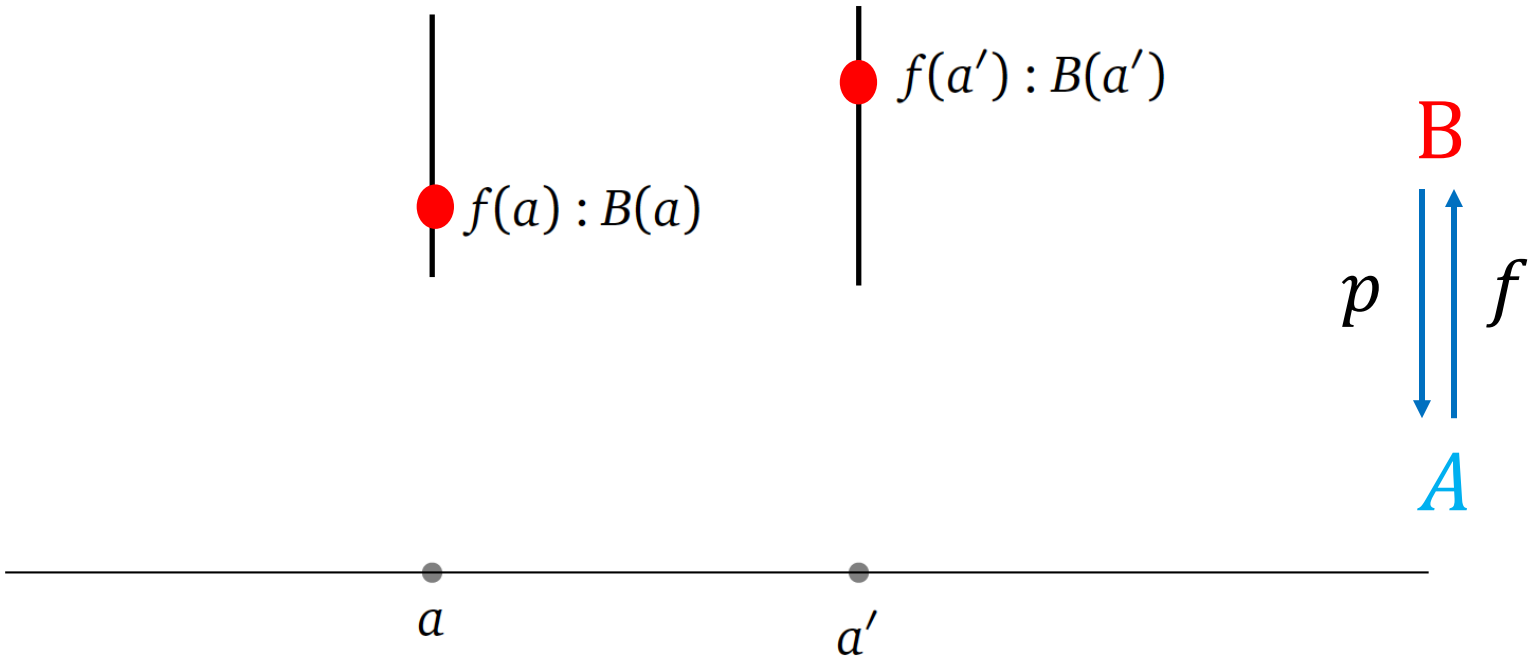
sがpのsectionである時($s: B \to E$)、それは、fiber上の「点」$s(x)$として表現されます。

Dependent Type
$\Pi(x{:}B).E(x)$ の項
$s(x) : \Pi(x{:}B).E(x)$
は、fiber上の「点」
section である。



すこし、名前を変えてみた
Dependent Typeの項のHoTTでの表現

$f(x)$: $\Pi(x{:}\,A).\,B(x)$



Grothendieck constructionの意味をたとえ話で考える

Grothendieck construction が、数学的にどのような意味を持っているのかを説明することは、数学的には簡単ではありません。

ここでは、すこし回り道をして、こうした構成がどのような問題に有効なのかを、たとえ話で考えてみたいと思います。

考えるのは、「語の世界」と「現実の世界」の素朴な意味のモデルです。

「語の意味」の素朴なモデル

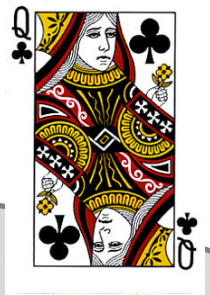
語の世界

WOMAN

QUEEN

MAN

KING



現実の世界

ものに「名前」をつける

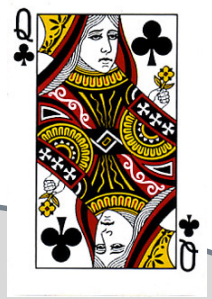
語の世界

WOMAN

QUEEN

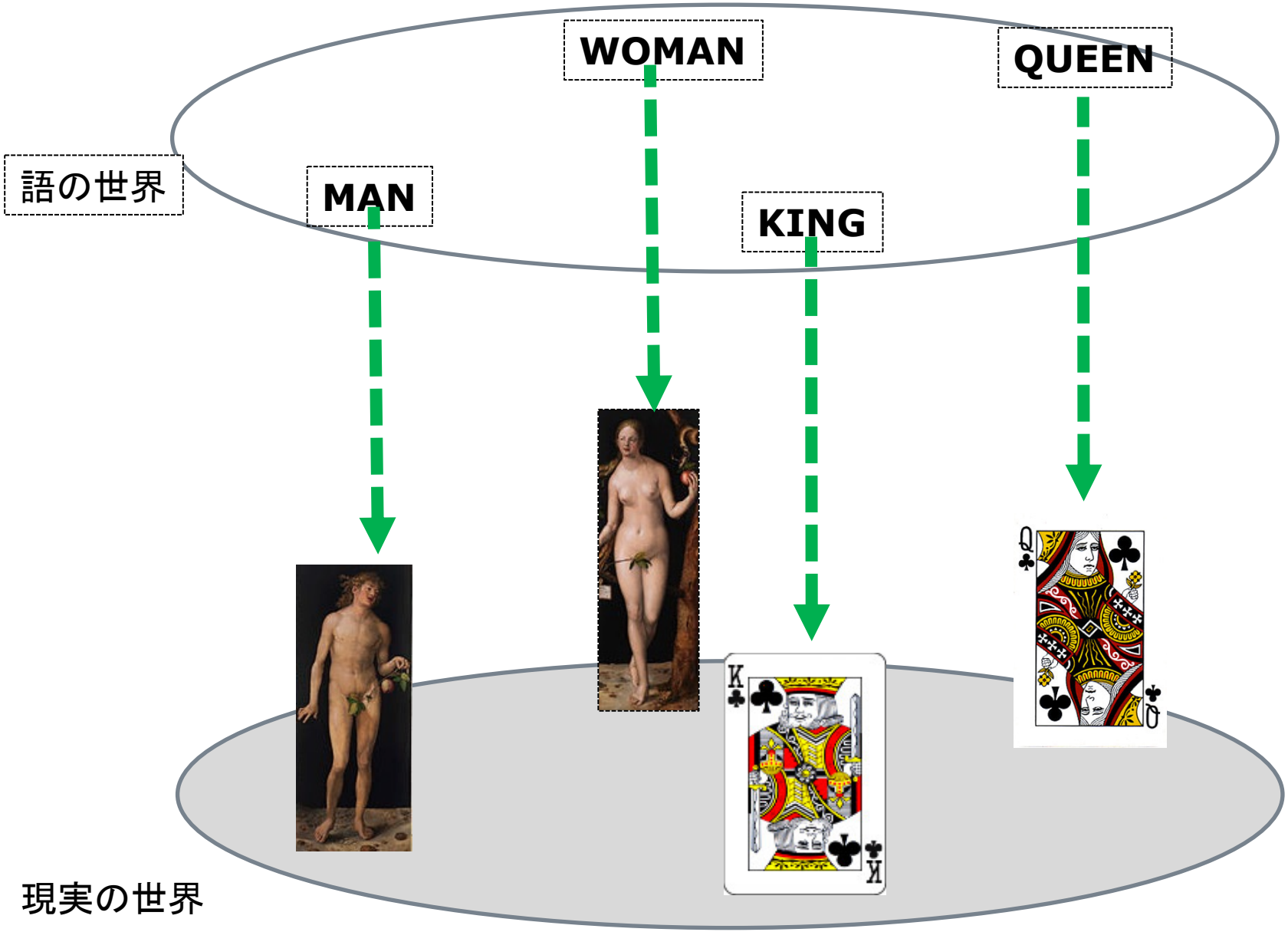
MAN

KING



現実の世界

語はものを指し示す



語の世界

日本語

自動車

飛行機

鳥

馬

英語

Automobile

Airplane

Bird

Horse

フランス語

automobile

avion

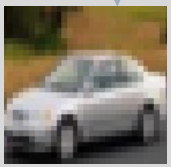
Oiseaux

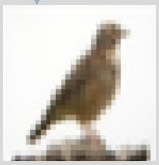
cheval

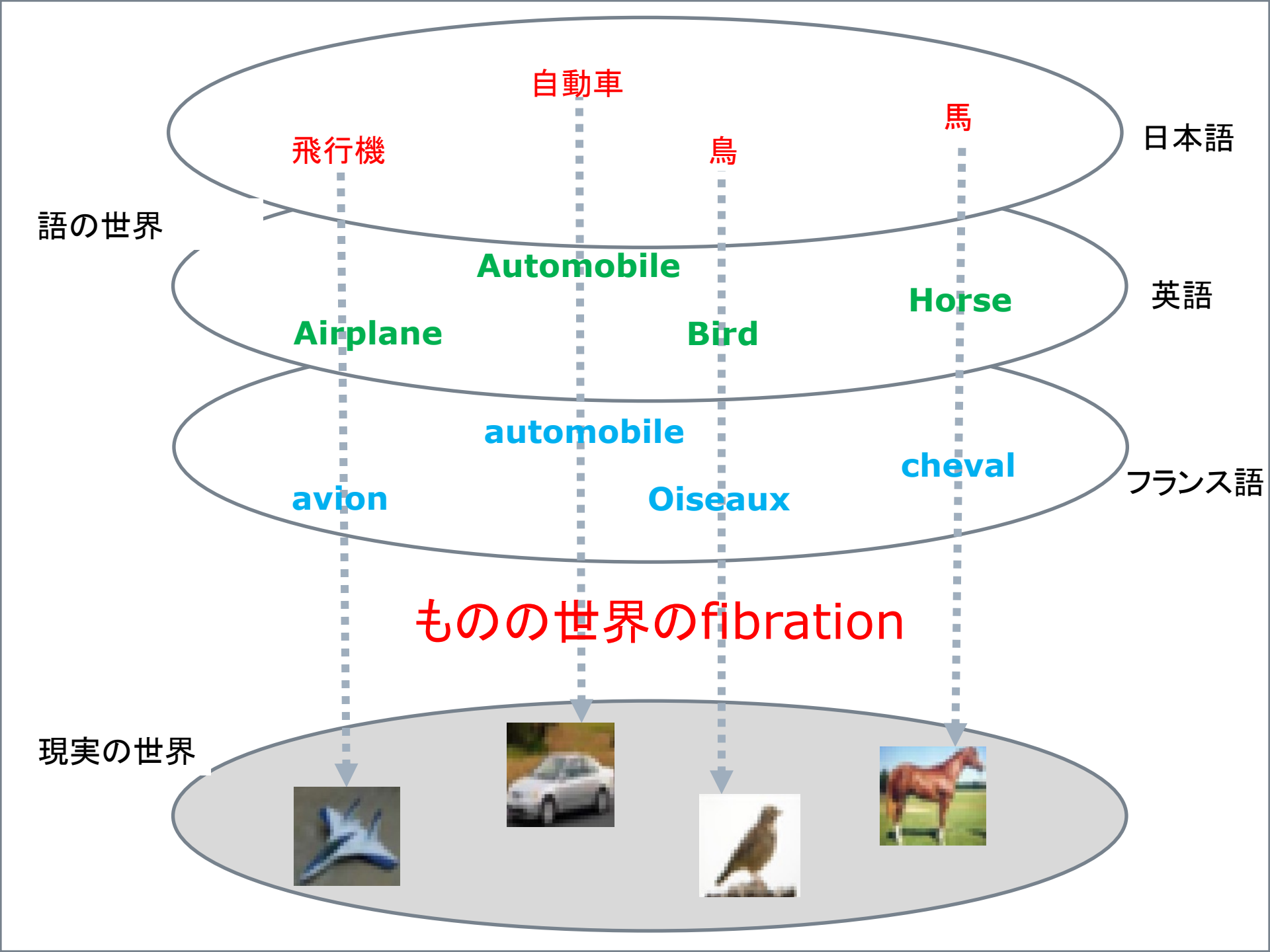
語はものを指し示す

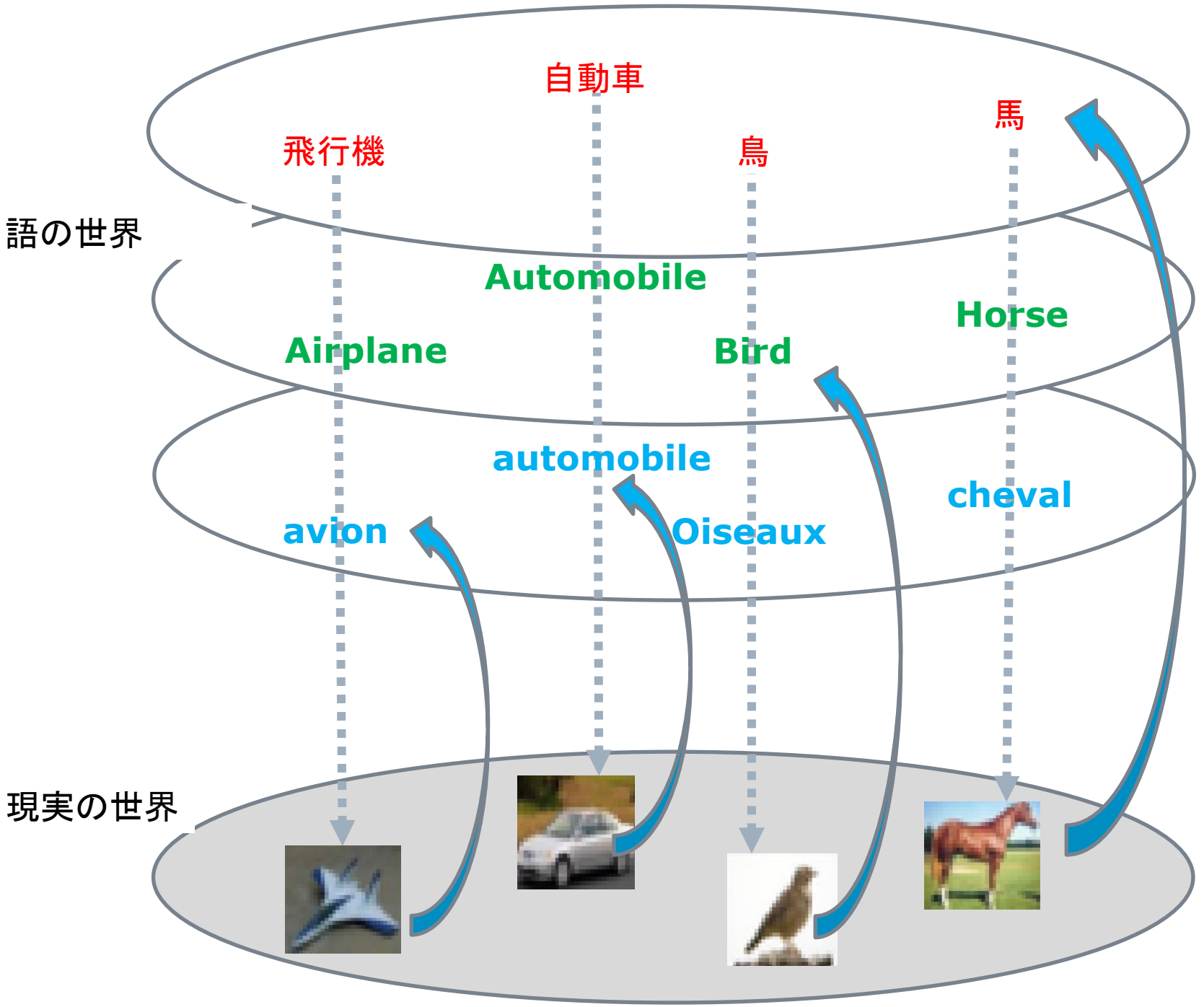
そうした関係は
言語が異なって
いても同じである。

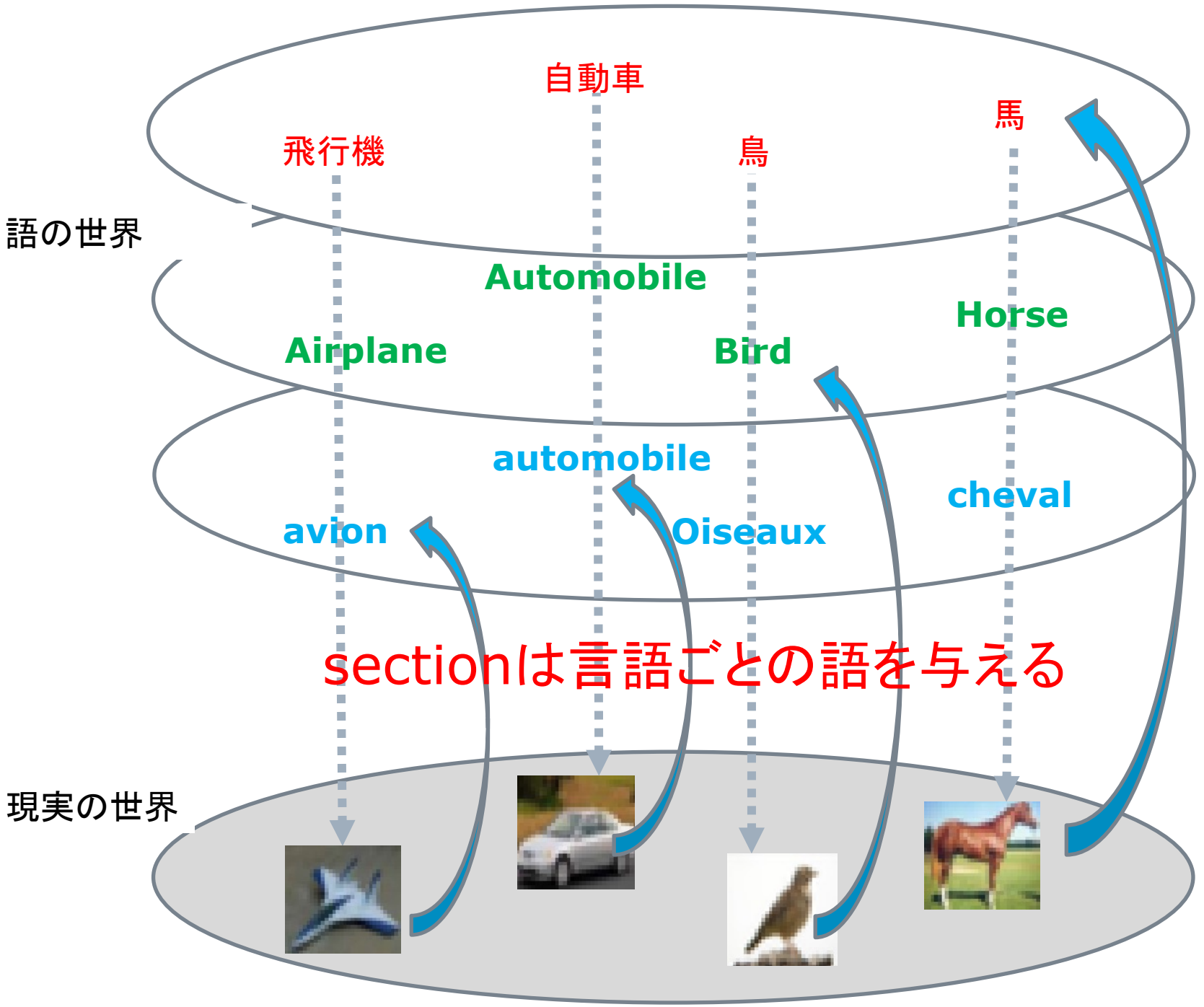
現実の世界

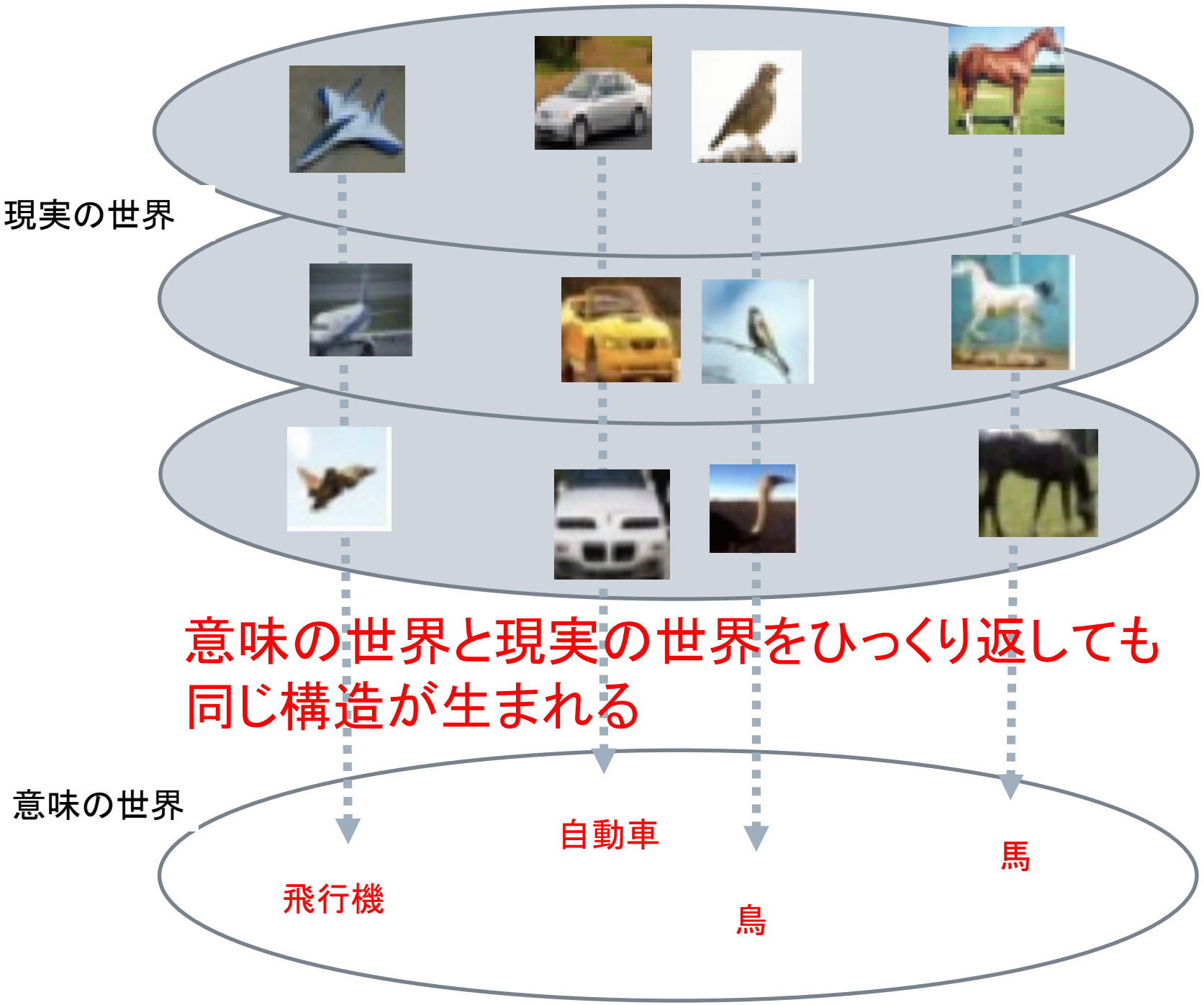


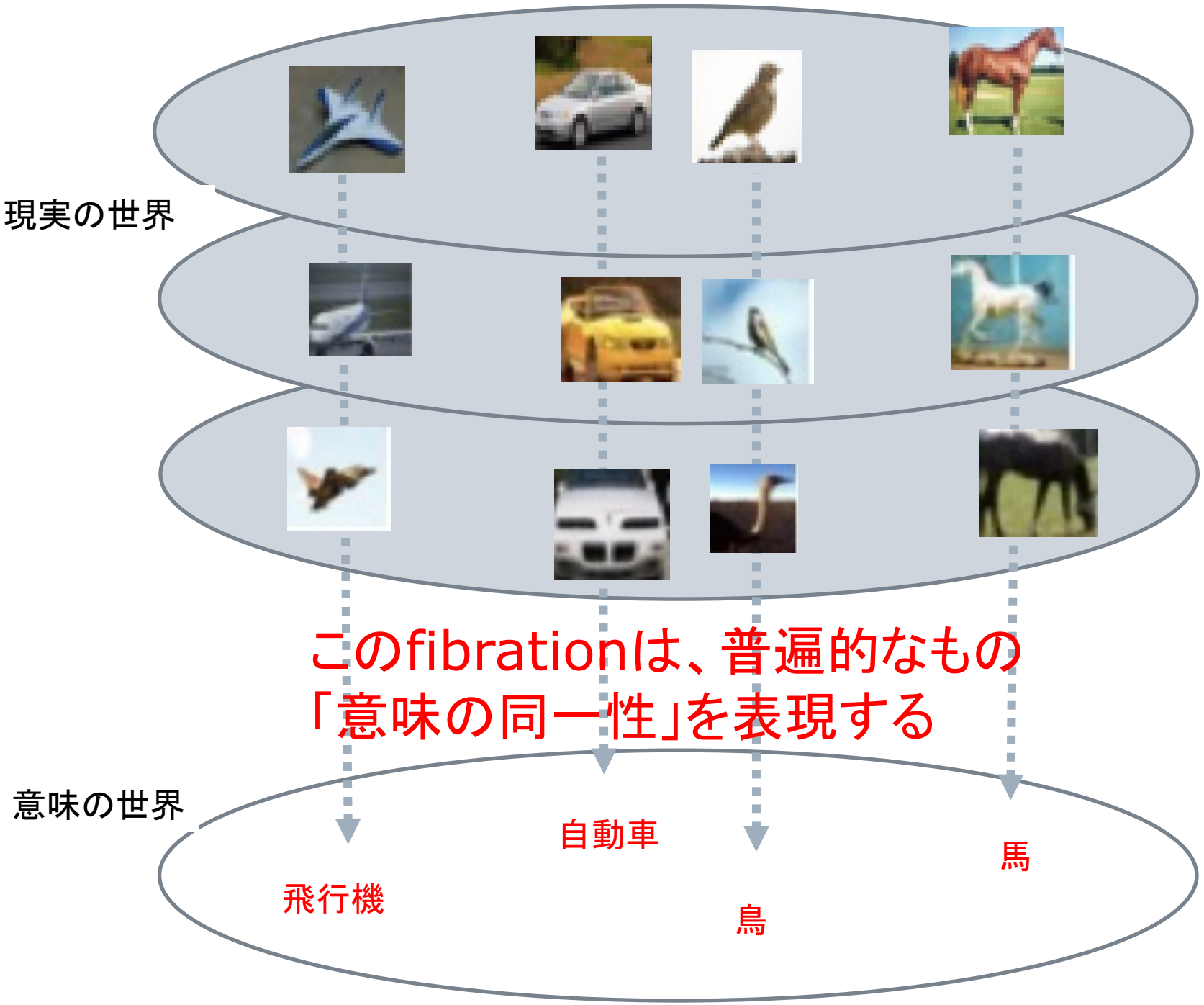


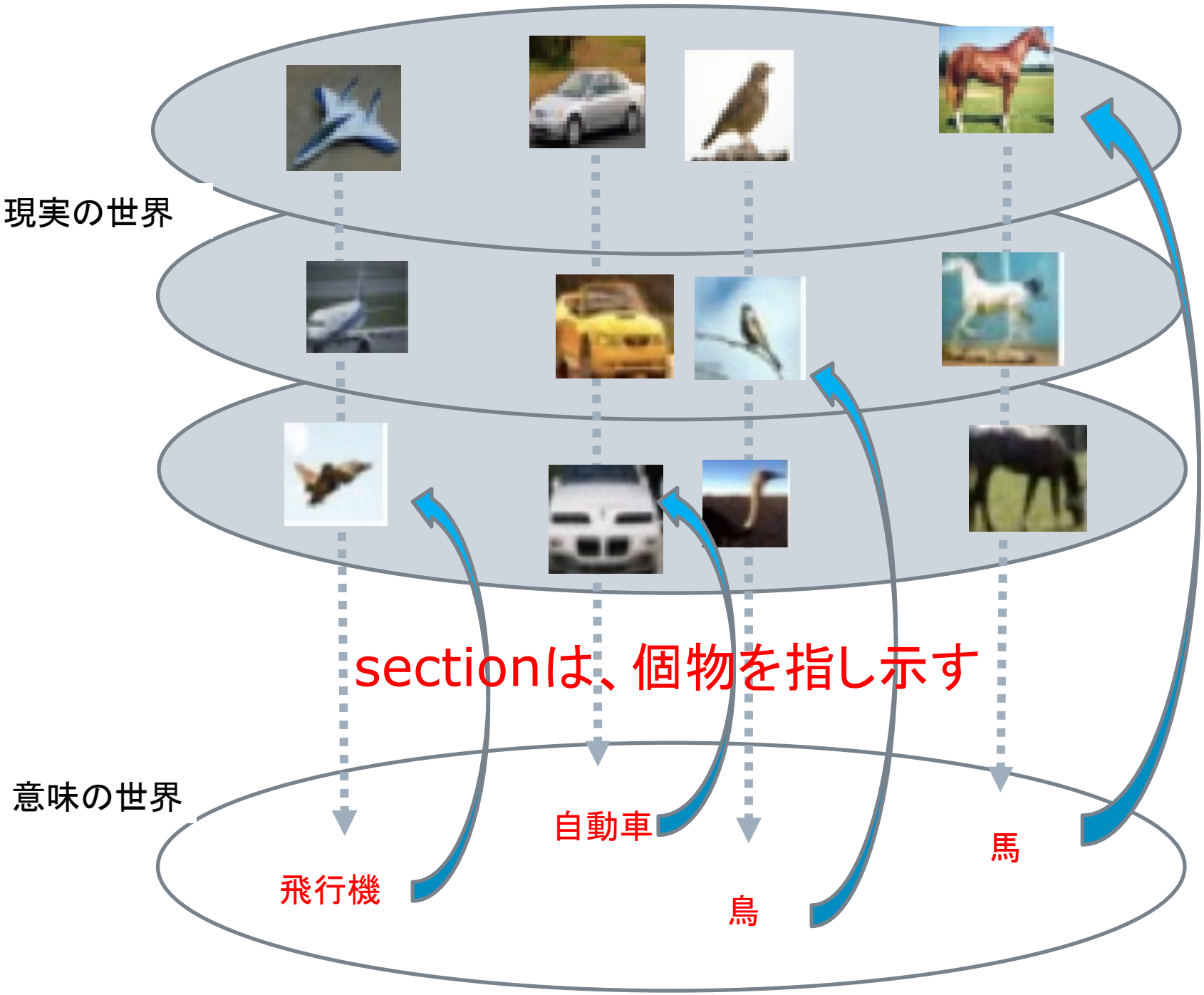

自動車
飛行機
鳥
馬
日本語
語の世界
Automobile
Horse
英語
Airplane
Bird
automobile
cheval
フランス語
avion
Oiseaux
ものの世界のfibration
現実の世界


語の世界
自動車
飛行機
鳥
馬
Automobile
Airplane
Bird
Horse
automobile
avion
Oiseaux
cheval
現実の世界


語の世界
自動車
飛行機
鳥
馬
Automobile
Airplane
Bird
Horse
automobile
avion
Oiseaux
cheval
sectionは言語ごとの語を与える
現実の世界


現実の世界
意味の世界と現実の世界をひっくり返しても
同じ構造が生まれる
意味の世界
飛行機
自動車
鳥
馬


現実の世界
このfibrationは、普遍的なもの
「意味の同一性」を表現する
意味の世界
飛行機
自動車
鳥
馬


現実の世界
sectionは、個物を指し示す
意味の世界
飛行機
自動車
鳥
馬

Tai-Danaeのcopresheaf意味論

Tai-Danaeのcopresheaf 意味論も、fibrationを利用しています。意味のcategoryの構成に用いられるL(x, −) は、x上のfibrationを考えることに他なりません。

Hom(x, −) でcopresheaf を、Hom(− , x) でpresheaf を構成することは、Grothendieck constructionのもっとも基本的な例です。

Tai-Danaeの議論については、マルレク「**大規模言語モデルの数学的構造**」を参照ください。

https://www.marulabo.net/docs/llm-math/

意味のcategory M の形

意味は、言語のcategory L のオブジェクト x から、Lの射 x → y によってうつされるすべてのオブジェクト yの集合によって表現されます。

Lのオブジェクトxを、その意味を表すcategory Setのあるオブジェクトに割り当てる functor を L(x, −)で表すと、

$$L(x, -) : L \to Set$$

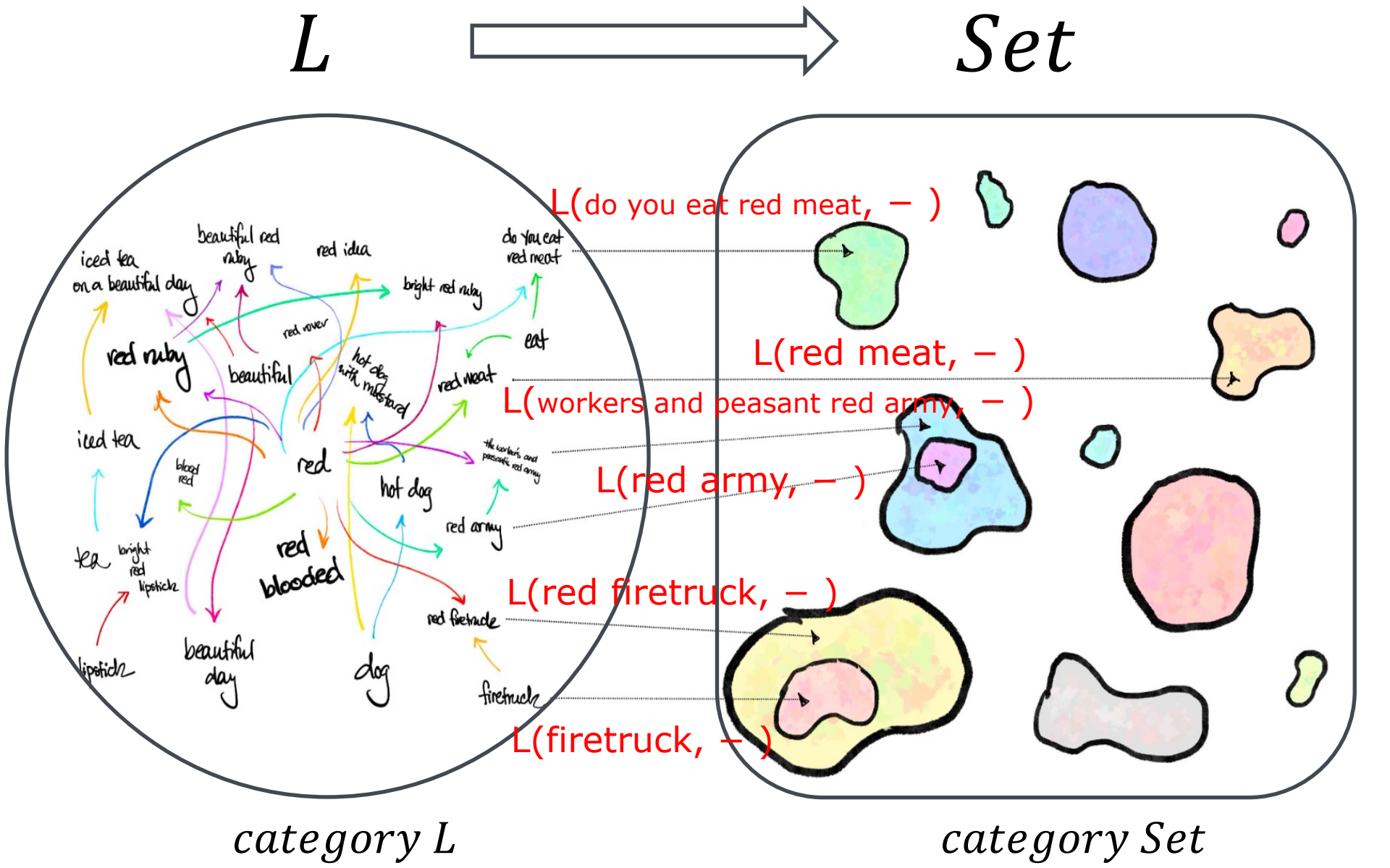
と表すことができます。

意味のcategory M のオブジェクトは、この

$$L(x, -) : Set^L$$

に他なりません。

Set^L のオブジェクト $L(x, -)$ の例



Homotopy Type Theory II
Univalent foundations

Homotopy Type Theory での「同一性」の扱い

このセッションで、Homotopy Type Theoryでの「同一性」の扱いを見ていこうと思います。

Dependent Type Theory では、「等しい」ことを表すのに二つの異なる表現を用いていました。

1. 二つの表現式が、同じ型αの内部で等しいこと
   $a =_\alpha b : \alpha$
2. 二つの型が、等しいこと
   $\alpha = \beta : Type$

Homotopy Type Theory でも、この二つの区別「同じ型に属するもの（項）が同じであること」と「型そのものが同じ」であることの区別は、引き継がれます。

Homotopy Type Theoryでは、素朴には、「型」は「空間」として、「項」は「空間」上の「点」として解釈されます。この解釈のもとで、二つの「同じ」はどのように解釈されるのでしょうか？

（この「解釈」は、たとえであって、数学的なものではありません。今回のセッションは、むしろこうした素朴な解釈が引き起こす問題をVeovodskyがどう考えていったかが一つの焦点になります。Univalence Axiomの導入。）

HoTT での素朴な「同一性」の解釈
aとbを結ぶ「道」がある時、aとbは「等しい」

以下、Homotopy Type TheoryをHoTTと略すことにしましょう。

HoTTでは、ある空間A内の二つの点 a, b が与えられた時、aとbを結ぶ「道」がある時、aとbは「等しい」と考えます。

この素朴な解釈のもとで、「同一性」が満たすべき次の基本的な条件が満たされることを、まず、見ておきましょう。

1. 反射律 (reflexivity): a = a
2. 対称律 (symmetry): a = b なら b = a
3. 推移律 (transitivity): a = b で b = c なら a = c

反射律
a ＝ a

ある点a から、自分自身に帰る道を考えます。

これは、a ＝ a であることを意味します。

先の解釈のもとで、反射律が成り立っています。

a ＝ a

自分自身に
帰る道

対称律
a ＝ b なら b ＝ a

aからbに向かう道があれば、その道を逆に辿ることができます。

これは、a ＝ b なら b ＝ a を意味します。

先の解釈のもとで、対称律が成り立っています。

a ＝ b → b ＝ a

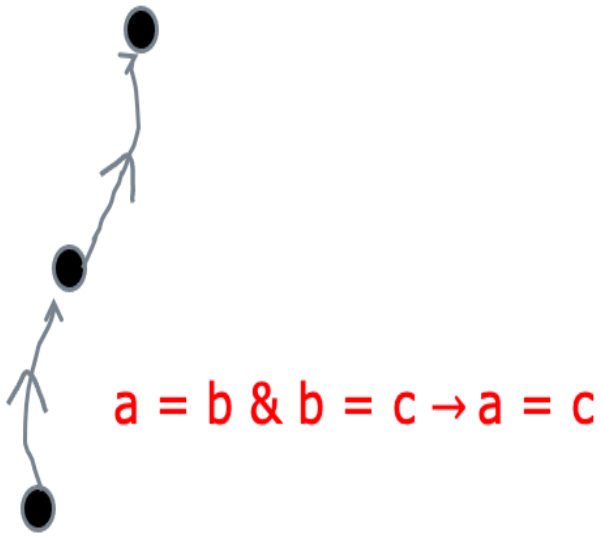
逆の道

推移律
a ＝ b で b ＝ c なら a ＝ c

a ＝ b はaからbに向かう道があることを意味し、b ＝ c は bからcに向かう道があるということですので、二つの道を結んだものも一つの道と考えれば、aからcへの道もあることになります。

これは、a ＝ b で b ＝ c なら a ＝ c を意味します。

先の解釈のもとで、推移律が成り立っています。



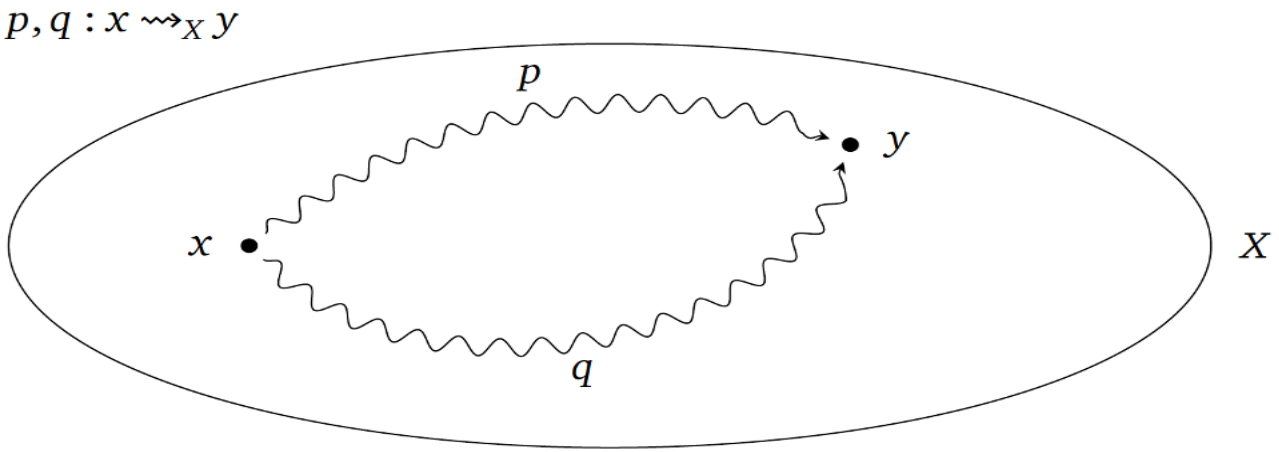
Homotopy とは何か？

空間Aの中で、点aと点bをつなぐ「道」がある時、
aとbは、同じものと見なす。

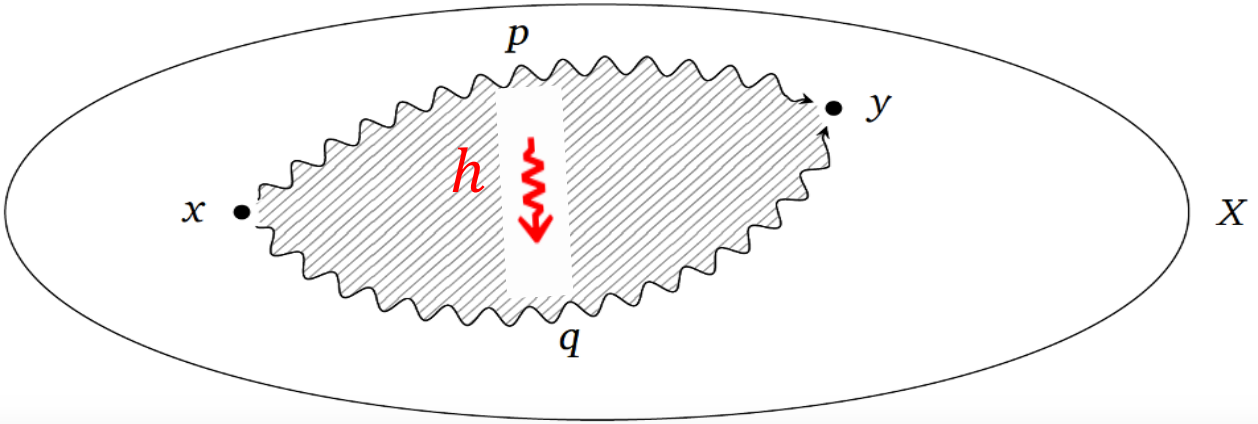
ということで、aとbとの「同一性」を見てきたのですが、点aと点bをつなぐ「道」自体は、連続的に変化しうるものです。次の図を見てください。

連続的に変化する「道」の「同一性」をあたえるものが、
homotopyなのです。

空間Xの二点 xとy を結ぶ二つの道p,qがあるとします。
これを$p, q : x \rightsquigarrow_X y$ と表します。



この時、pとqを同一視する道pから道qへの「道」$h : p \rightsquigarrow_{x \rightsquigarrow y} q$
を、homotopy と呼びます。

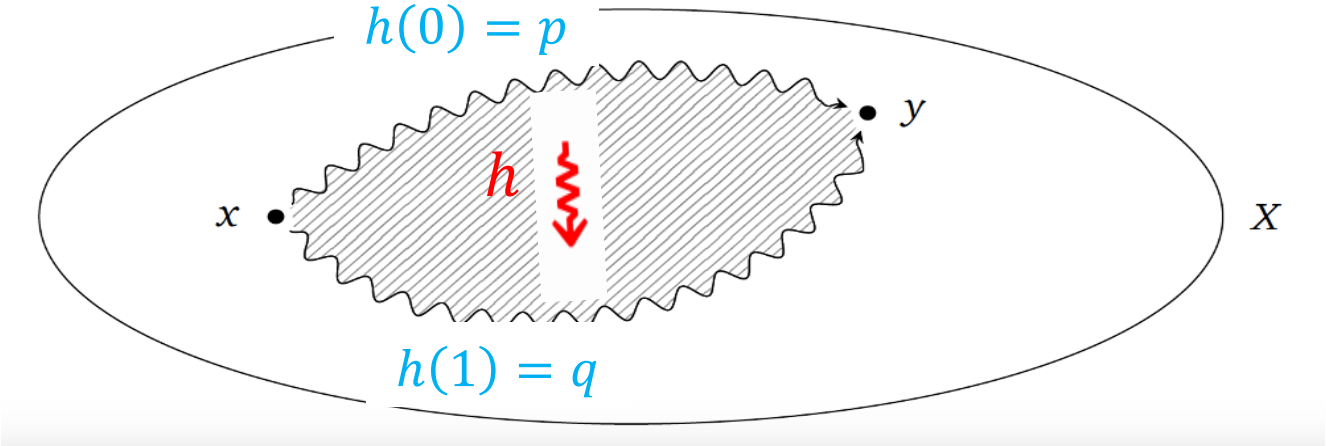


homotopy h は、xからyへの連続関数pを、xからyへの連続関数qに、連続的に変化させます。

homotopy h は、$0 \leq t \leq 1$の範囲で変化するパラメータ tを持っていて、

$t = 0$ の時、$h(0)$は関数pに一致し、

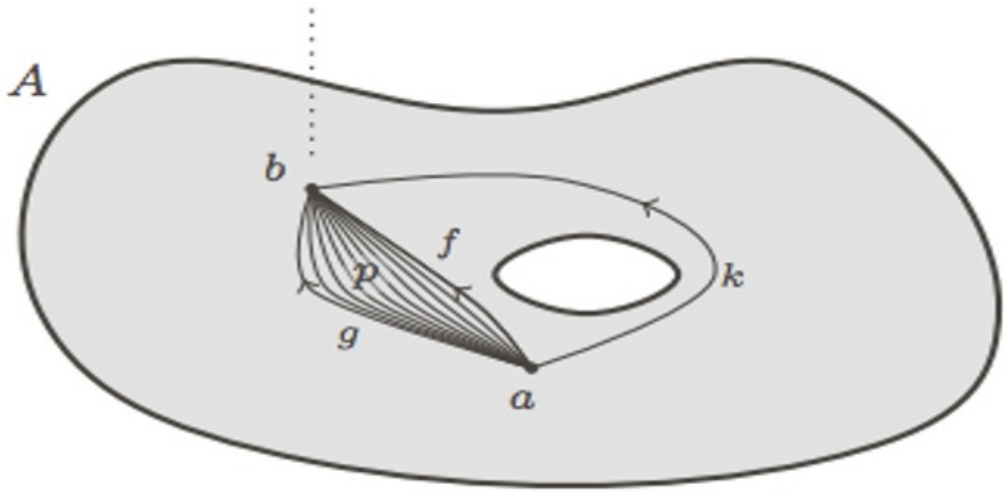
$t = 1$ の時、$h(1)$は関数qに一致します。



少し複雑な「空間」上の「道」の例

「道」とその「同一性」であるhomotopy が定義される「空間」は、単純なものばかりとは限りません。

次のような例で、「道」とそのhomotopyを考えてみましょう。



この図には、aからbに向かう道がいろいろ書き込まれているのですが、「穴」左側のgとかfと、「穴」の右側を通るkとは、違う特徴を持つことに注意してください。

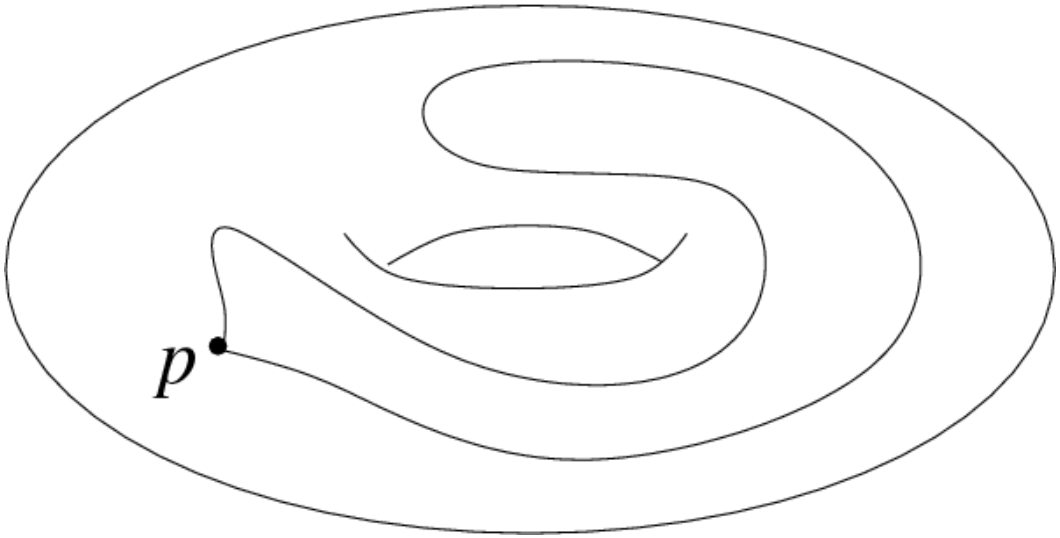
gとかfは、連続的な変形で一つのhomotopy pで代表されるのですが、gとかfは、「穴」に邪魔されて連続的変形によっては、kに移ることはできません。

逆に、kは一本しか書かれていませんが、この空間Aには、kとhomotopy（例えばqで）同値となる仲間がたくさんいるはずです。

ここでは、具体的なf,g,..,kについて話したのですが、この空間Aの二点を結ぶ異なるhomotopyは、少なくとも二種類あることはわかります。

こうした性質は、具体的な f,g,.,k の選択にはよらない、空間X自身のtopologicalな性質によるものです。(位相空間の「基本群」と呼ばれるものです。)

先の例では、空間Aの二つの点を結ぶ「道(path)」を考えましたが、Aから一つの点 a を取り出して、その道 – aからaに向かうループ -- を考えても、ある空間では異なるhomotopy同値のループがありうることがわかります。



Contractible なループ

先の図のpから出て、ドーナツの中身の部分を走って、自分自身であるpに戻ってくるループは、面白い性質を持っています。

このループとhomotopy同値のループは、どんどん小さなループに縮んでいって、最後には点pを残して消えてしまうループを含んでいます。

こうしたループを「contractibleなループ (縮退可能なループ)」と呼びます。

Contractibleな型の定義

HoTTでは、「縮退して点だけ残してなくなってしまう」
contractibleなループは、大事な意味を持っています。

型Aがcontractibleであることを、次のような項を構成できることと定義します。

$$isContr(A) :\equiv \sum_{x:A} \prod_{y:A} y \rightsquigarrow x$$

この定義は次のように読みます。

型Aのすべての項yについて、yから型Aの項であるxへの
path が存在するなら、型Aはcontractibleである。

Contractibleな型の性質

Contractibleな型は、次のような性質を持っています。

- この型（空間）には、一つの項（点）が存在していて、任意の点からこの点へのpath が存在する。
- Contractibleな型は、一つの項（点）のみを含むので、singleton typeと呼ばれる。
- Contractibleな型に属する任意の二つの項（点）の間には、pathが存在する。

型は空間で項は点なのか？

だんだん議論が抽象的になってきて、「型は空間で項は点である」という当初の素朴な解釈から離れて行っているのを感じます。

Veovodskyは、そもそも位相空間を一つの型として捉えようとしてHoTTを構築したのでしょうか？ そうではないと思います。というのも、カテゴリー論的には、位相空間は無限の階層をもつきわめて複雑な対象だからです。

Voevodskyがしようとしたことは、数学的に正確に言えば、型の理論を「Kan complexのカテゴリー(抽象的なSimplicial complex)」で解釈しようとしたのです。

ただ、このカテゴリーと位相空間のカテゴリーの間には、”Quillen equivalence”という「等価性」があります。ですので、「型は（位相的）空間である」という直観的な理解は、正当化できるのです。

このあたりの理論的背景については、今回のセミナーでは省略しますが、興味がある方は、次の論文を参照ください。

“The Simplicial Model of Univalent Foundations (after Voevodsky)”
https://arxiv.org/abs/1211.2851v5

型の「同一性」をどのように示すのか？

これまでみてきた、homotopy同値による「空間」A内の「点」 a, b の「同一性」の定義は、型の理論で言えば、同じ型Aに属する項 a, b の「同一性」の定義でした。

それは、次のタイプの定義でした。

- 二つの項 a, bが、同じ型Aの内部で等しいこと
  $a =_A b : A$

ただ、従属型理論には、次のタイプの「同一性」の定義があります。

- 二つの型が、等しいこと
  $\alpha = \beta : Type$

HoTTでは、この第二のタイプの同一性、そもそも型αと型βは等しいことをどのように定式化すればいいのでしょう？

奇妙なことに気づきます。第一のタイプの同一性の定式化で有効だった、ある空間での点の同一性をhomotopy同値とfibrationで追求するというアプローチは、この問題ではあまり有効には見えません。なぜなら、ここでの問題の焦点は、点ではなく空間そのものの同一性にあるからです。

無限の階層を持ち、複雑に分岐した構造を持つ位相空間そのものを対象にし、それを構成する任意の空間に自明でない「同一性」あるいは「非同一性」を導入することは、不可能に思えます。

Univalence Axiomの発見

Voevodskyが行ったことは、実は、Homotopy Typeの世界にこれまでの数学の延長上に守備よく型の同一性を導入することに成功したということではないのです。

彼が考えたことは、それが極めて困難なことには何か理由があるからだということです。それは、我々がさまざまな数学の基礎である同一性について何か基本的なことに気づいていないという反省でした。

彼は、今まで誰も気づいていなかったスタイルで、同一性についてのある命題を提出し、これを公理として認めれば、Homotopy Typeの世界でも「型の同一性」を導くことができることを示しました。

その公理を”Univalence Axiom”と言います。公理ですから、証明ができるものではないことに、注意してください。

それればかりではありません。

彼は、このUnivalence Axiomを含む公理系をベースにして、数学の基礎を見直すこと、この公理系を基礎として全数学を再構成することを提案します。こうした動きをUnivalent Foundationsと言います。

以前に紹介した数学の証明にコンピュータを使おうという
UniMathプロジェクトは、まさにこうしたUnivalent
Foundationsの取り組みに他なりません。

それは、これまでの既存の枠組みの数学を、コンピュータ上に移
植しようとするプロジェクトと、その目的においては異なるものです。

もっとも、数学の基礎にかかわる部分を除いては、両者は同じ能
力を発揮すると考えていいと思います。

Univalence Axiomとは何か？

Univalence Axiomについては、数学的には語るべきことはたくさんあるのですが、それについては別のセミナーをするのがいいと考えています。

今回のセッションでは、Univalence Axiomの数学的な背景説明を省略しました。というのも、Univalence Axiomは、数学的な主張ではありますが、「同じとは何か？」といういわば古くからの認識論的・哲学的な問題についての問題の整理を含んでいます。

多くの数学者が、彼の問題提起を歓迎したのは、数学者としての経験に照らして、それが問題の見事な整理になっていると感じたからだと思います。

ここで紹介するのは、数学者のSteve Awodeyが、おそらくは数学を専門としていない人を対象にUnivalence Axiomを説明した2014年の次の論文の丸山による概訳です。

とても優れた紹介だと思います。

"Structuralism, Invariance, and Univalence"

https://www.andrew.cmu.edu/user/awodey/preprints/siu.pdf

構造主義の原理

「同型な対象は同一である」
“Isomorphic objects are identical. “

これを「構造主義の原理 The Principle of Structuralism 」として、PSで表そう。

例

- コーシー列で定義された「実数$_{Cauchy}$」も、デディキンドの切断で定義された「実数$_{Dedekind}$」も同型である。解析的には、二つの実数は、同じものである。
- 単位区間 [0,1]は、区間 [0,2]と位相同型である。トポロジー的には、この二つの空間は、同じものである。

弱い形の構造主義の原理

この原理は、実践的には有用なのだが、集合論的には、正しくない。先の二つの例でも、集合としては、二つは同じものではない。

「同一性 Identical」を、もっと弱い形で表現してみよう。

AとBが同型ならば、関係するすべての性質Pについて、
P(A)が成り立つなら、P(B)が成り立つ。

AとBが同型であることを、$A \cong B$ と表そう。
この弱い形の原理をPS’としよう。

確かに、PS’の主張は、PSより弱い。しかし、「関係するすべての性質」というのは、何を指しているのだろう？

同型について

二つのものが同型であるというのは、どういうことだろう？
すぐに思いつくのは、次の定義である。
以下、AとBが同型であることを、$A \cong B$で表す。

$A \cong B \iff$ AとBは同じ構造を持つ

ただし、これは、「同じ構造を持つ」ということの定義であって、「同型」の定義ではない。

カテゴリー論での同型の定義

カテゴリー論では、次の条件を満たす時に、AとBは同型であるといわれる。

構造を保存するような写像 f, gがあって、
$f: A \to B$ と $g: B \to A$ が、$f \circ g = 1_A$, $g \circ f = 1_B$を満たす。

構造的性質と不変性

ある性質P(X)は、次の性質を満たす時、不変であるという。

$$A \cong B \ \& \ P(A) \Longrightarrow P(B)$$

この不変性は、この同型に関しての不変性で、構造的性質ともいわれる。

もし、$A \cong B$ で、P(X)が構造的性質で
P(A)が成り立つなら、P(B)が成り立つ

この性質は、実践的には役に立つのだが、同型について何が構造的性質かは、この定義は何も語っていない。

構成的型の理論と、Curry-Howard対応

構成的型の理論では、基本的なオブジェクト、すなわち、個物の型 Xは、次のものである。

$$0,\ 1,\ A+B,\ A\times B,\ A\to B,\ \Sigma_{x:A}B(x),\ \Pi_{x:A}B(x),\ \mathsf{Id}_A(x,y).$$

そして、それらは次の論理的な命題に対応している。

$$\bot,\ \top,\ A\vee B,\ A\wedge B,\ A\Rightarrow B,\ \exists x:A.\ B(x),\ \forall x:A.\ B(x),\ x=_A y.$$

こうした対応は、Curry-Howard対応、あるいは、
“Propositions-as-Types” と呼ばれる。

$$\text{proof} : \text{Proposition} \quad \approx \quad \text{term} : \text{Type}$$

すなわち、ある命題の証明は、対応する型の項になる。

不変性の原理

型の理論は、オブジェクトのすべての定義可能な性質は、不変であるという重要な性質を持っている。

もし、Pが、ある基本的な型の上で定義可能なら、次の推論は演繹可能である。

$$\frac{A \cong B \qquad P(A)}{P(B)}$$

これを、型の理論の不変性の原理という。すなわち、

すべての定義可能な性質は、同型のもとで不変である。

オブジェクトの同一性（Identity）

オブジェクトAとBが、同一であるというのはどういうことだろうか？

型の理論では、同一性を、ライプニッツの法則で定義する。

$$A = B \;:=\; \forall P.\; P(A) \Rightarrow P(B),$$

この同一性は、AとBが共通に属する同じ型 Xの項の同一性である。$\forall P.$ は、X上のすべての性質の型である$\mathcal{P}(\mathrm{X})$上（Pの部分集合である）を走る。

だから、正確には、先の式は次のようになる。

$$A =_X B \;:=\; \forall P : \mathcal{P}(X).\; P(A) \Rightarrow P(B),$$

これは、オブジェクトの間の同一性を示すもので、型Xと型Yの同一性を表すものではない。

型の同一性

構成的型の理論には、すべての型Xの項について、基本的な同一性の関係 $\mathrm{Id}_X(a, b)$が存在するので、項の同一性については、次の推論規則が成り立つ。

$$\frac{\mathrm{Id}_X(a,b) \qquad P(a)}{P(b)}$$

それでは、型の同一性については、どのような推論が可能だろうか？　それには、すべての型のUniverseであるUを追加すればいい。

$$\frac{\mathrm{Id}_U(A,B) \qquad P(A)}{P(B)}$$

型の同一性と同型

型の同一性の推論の仕方はそれでいいとして、型の同一性と同型の関係はどうなるのか考えてみよう。

Uを追加する前に、定義可能なすべての性質は不変であることを見てきた。すなわち、

$$\frac{A \cong B \qquad P(A)}{P(B)}$$

不変性の原理

もしも、Uを含んだPについても、この推論が成り立つのなら、

$$P(X) := \mathsf{Id}_U(A, X),$$

と置けば、$\mathsf{Id}_U(A, A)$ から、次の構造主義の原理を導くことができる。

$$A \cong B \;\Rightarrow\; \mathsf{Id}_U(A, B)$$

すなわち、型の理論では、不変性の原理は、構造主義の原理を含意する。

簡単に言えば、Universeを持つ拡張された型の理論でも、すべての定義可能な性質は、同型不変である。この時、特別のオブジェクトは、同一のものになる。

こうしたことは、本当に可能なのだろうか？

「… である」の意味するもの
型と命題の同一視　Curry-Howard対応

型の理論では、すべての命題がある型（すなわち、その命題の証明の型）に対応する。そして、すべての型は、命題（すなわち、その型が項を持つという命題）とみなすことができる。

例えば、A ≅ Bという命題には、AとBとの同型の型 $\mathsf{Iso}(A, B)$ が対応する。（以下で X ≈ Yは、「XとYは対応する」を表す）

$$\text{“}A \cong B\text{”} \approx \mathsf{Iso}(A, B)$$

同様に、$\mathsf{Id}_U(A, B)$ という型には、AとBは同一の型であるという命題　$A =_U B$ が対応する。

$$\text{“}A =_U B\text{”} \approx \mathsf{Id}_U(A, B).$$

以下では、型の理論の慣例（ Curry-Howard対応）に従って、型と命題を同一視することにする。

$A \cong B$と$A =_U B$

同型関係は、反射的なので、次のような写像があることを示すことができる。

$$(A =_U B) \to (A \cong B),$$

もし、AとBが集合なら、後述のUnivalence Axiomを使えば、この写像自身が、同型であることを示すことができる。

$$(A =_U B) \cong (A \cong B).$$

これは、次のような写像が存在することを意味する。

$$(A \cong B) \to (A =_U B)$$

これは、「同型なものは、同じものである」という、構造主義の原理である。

等価性　Equivalence

Voevodsky は、射 f：A → B が、equivalent「等価である」という概念に、次のような単純で統一的な定義を与えた。

1. もし、AとB が集合であるなら、それは集合間の全単射（一対一対応のこと）である。
2. もし、AとB が命題であるなら、それは命題間の論理的等価性である。
3. もし、AとB がgroupoids であるなら、それはgroupoid間のカテゴリー的等価性である。

Voevodskyの Univalence Axiom

Voevodskyの Univalence Axiom は、先のA, Bが集合の場合の　$(A =_U B) \cong (A \cong B).$ をもっと一般的な形で述べたものである。

$$(A =_U B) \simeq (A \simeq B),$$

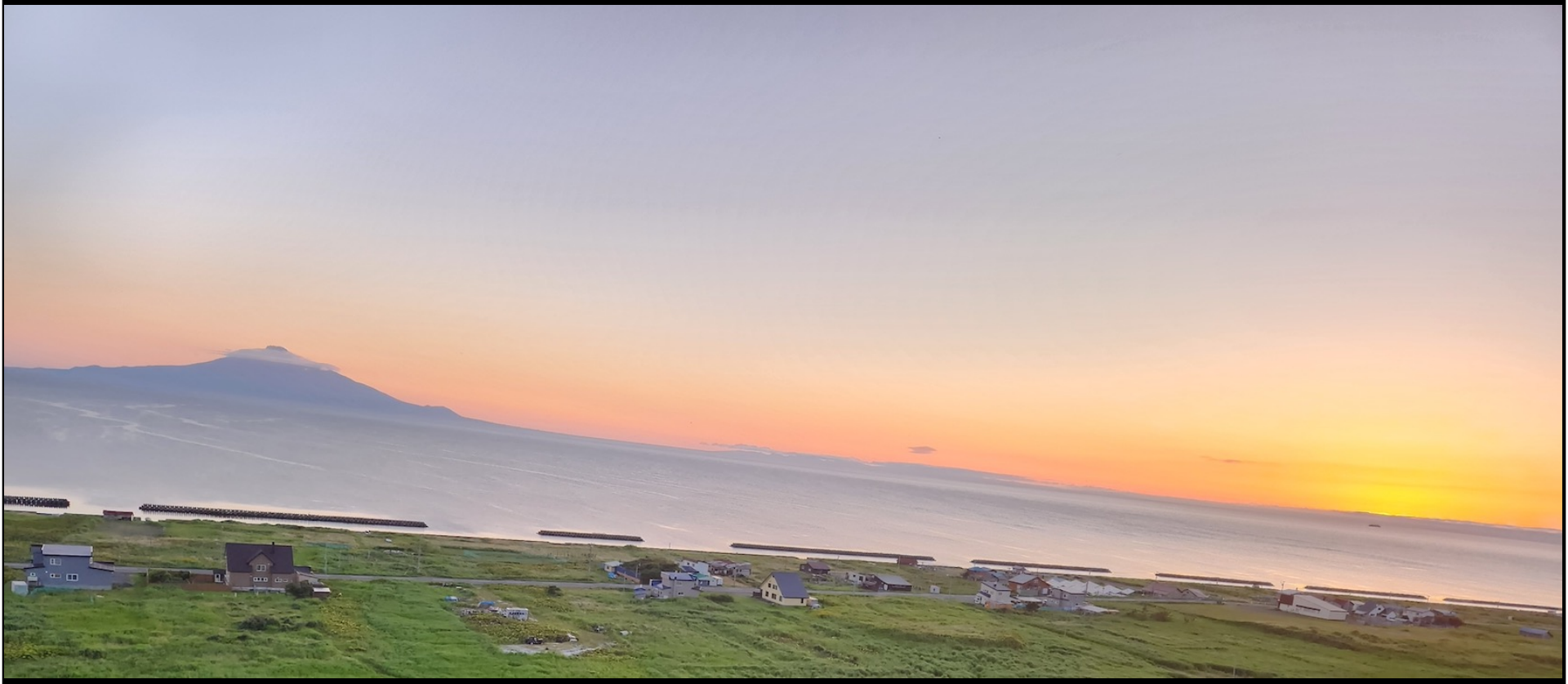
すなわち、

二つのものの同一性は、二つのものの等価性と等価である

二つの型A,Bの同一性は、A,Bの等価性と等価である。

Univalence Axiomから「同一性」についての多くの命題が導かれる

- 構造主義の原理
  - 二つの同型な集合は等しい
  - 二つの同型な代数的構造は等しい
- 数学でのUnivalence Axiomの利用
  - 二つの (カテゴリー的に) 等価なgroupoid は等しい
  - 二つの等価なカテゴリーは等しい
- ライプニッツの原理
  - aとbが等しいなら、aのすべての性質は、bの性質でもある



Part 3

コンピュータによる証明

Part 3 Agenda
コンピュータによる証明

連続体仮説の独立性証明

Feit–Thompson定理の形式的証明

ケプラー予想の証明

連続体仮説の独立性証明

Cantorが考えたこと
無限の無限の系列

可算無限集合の濃度を$\aleph_0$（アレフゼロ）と呼ぶ。

集合Aの濃度（要素の数）を|A|で表すとする。

$|\omega| = \aleph_0$ である。

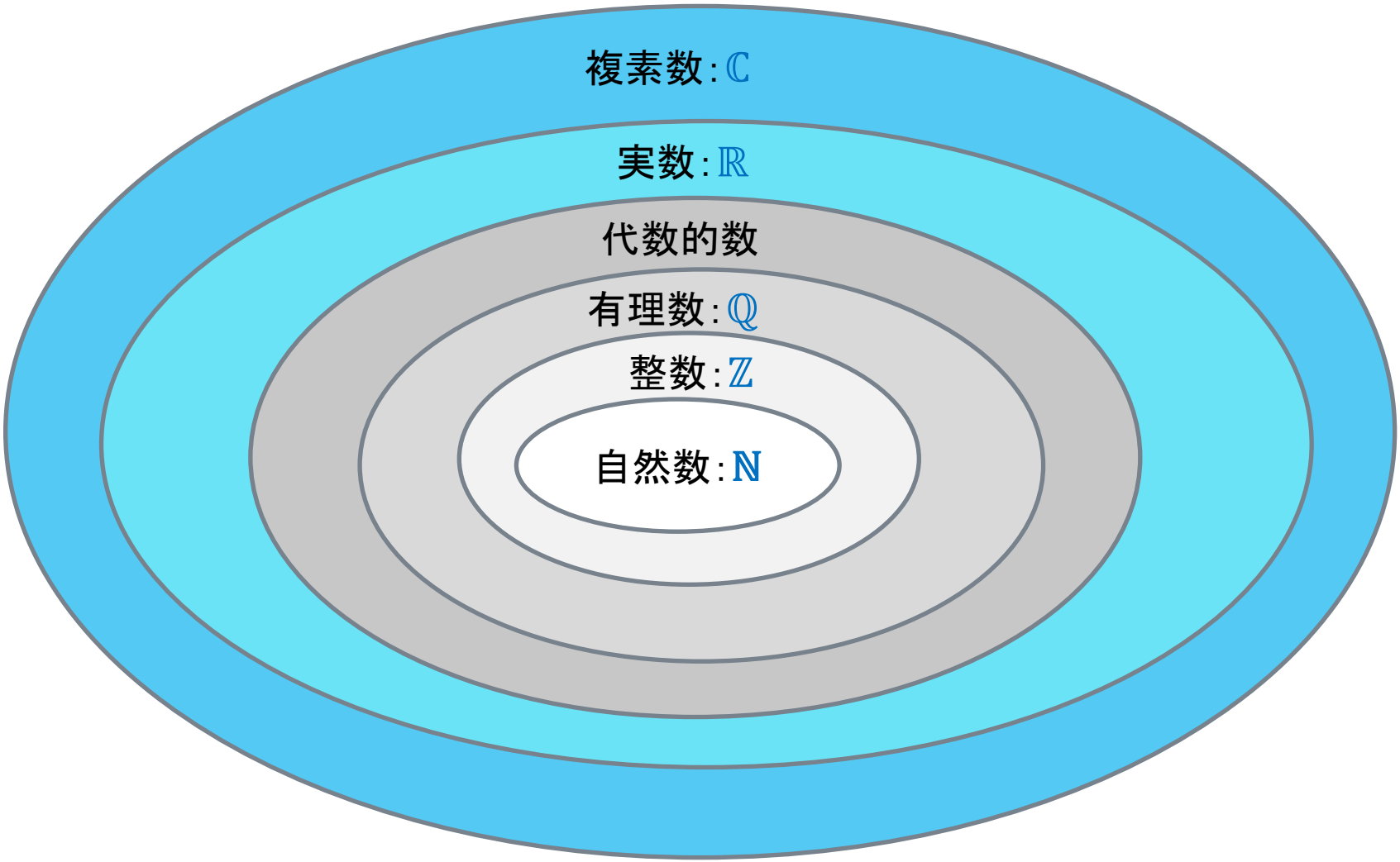
任意の濃度 $\aleph_\alpha$について、$\aleph_\alpha < 2^{\aleph_\alpha}$を示すことはできるので、無限の濃度にも無限の階層があることがわかる。

こうして、順序数と基数の二つの無限の系列があることになる。有限集合の時、両者は、ある自然数に一致する。

順序数の系列：$0,1,2,3,\ldots,\omega,\ldots$

基数の系列　：$0,1,2,3,\ldots,\aleph_0,\aleph_1,\aleph_2,\aleph_3,\ldots$

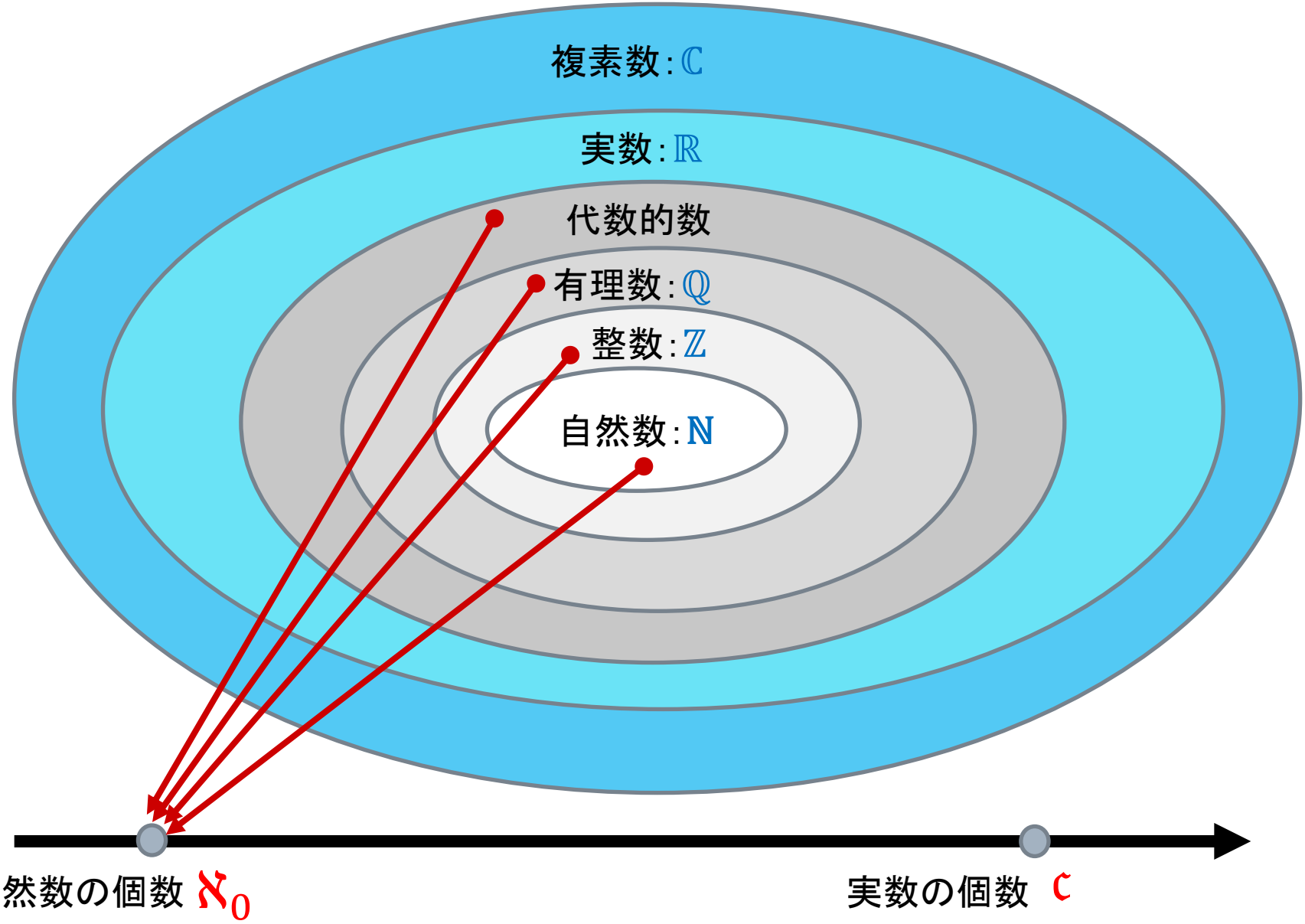
数の階層



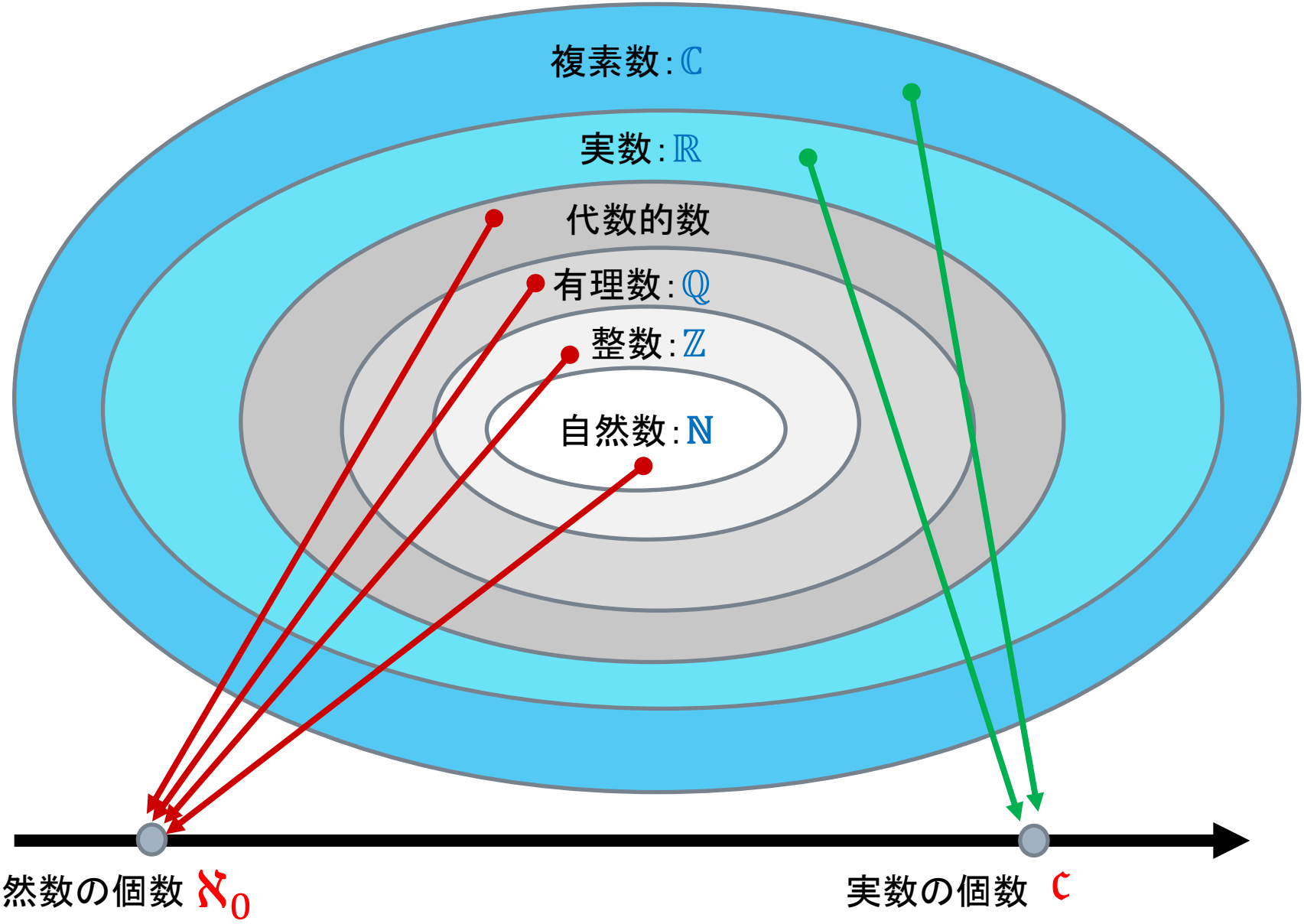
自然数の個数 $\aleph_0$

実数の個数 $\mathfrak{c}$

無限の数の濃度についてカントールが示したこと



無限の数の濃度についてカントールが示したこと



Cantorが解けなかった問題
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。

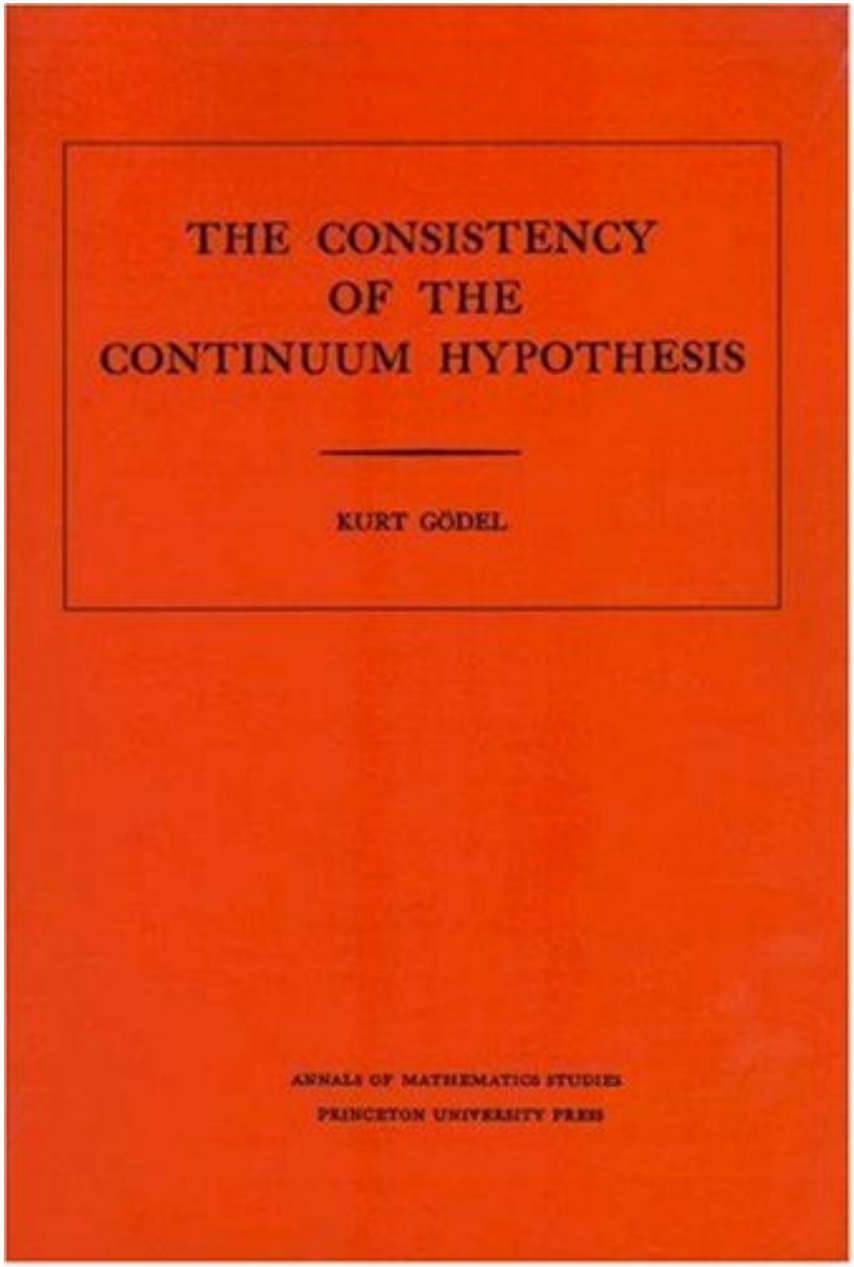
ヒルベルトの23の問題　1900年

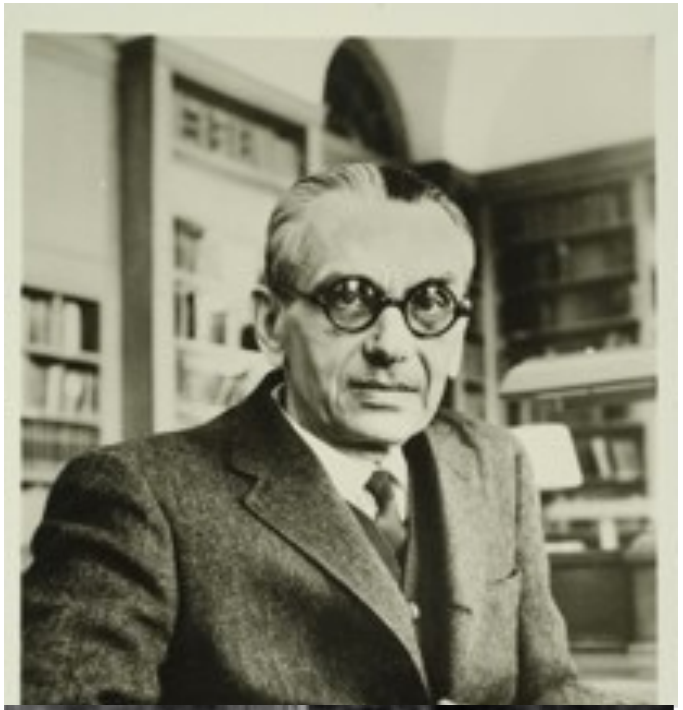
ヒルベルトは、1900年に、20世紀の数学が解くべき23の問題のリストを発表した。その問題提起は、20世紀の数学に少なくない影響を与えた。カントールの「連続体仮説」は、この23の問題の第一番目の問題に取り上げられている。

ゲーデル
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。


THE CONSISTENCY
OF THE
CONTINUUM HYPOTHESIS
KURT GÖDEL
ANNALS OF MATHEMATICS STUDIES
PRINCETON UNIVERSITY PRESS



ゲーデル 完全性定理 1929年
「無矛盾な理論はモデルを持つ」

ゲーデルは、ある形式的体系の中に、その証明もその否定の証明も、その形式的体系の中では与えることができない命題Pが存在するという「不完全性定理」（1930年）で有名である。特に、その形式的体系が「無矛盾」であることを主張する命題自身が、この形式的体系では証明できないという発見は、よく知られている。

ゲーデルには、もう一つ、重要な業績がある。それは「無矛盾な理論はモデルを持つ」というもので、「完全性定理」と呼ばれる。ある形式体系のある命題Pが、その形式的体系のすべてのモデルで妥当する時、その命題Pは証明可能となる。

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。

J.P.Cohen
1934-2007

[19] Gardner, R. S., A. J. Wahba, C. Basilio, R. S. Miller, P. Lengyel, and J. F. Speyer, these PROCEEDINGS, **48**, 2087 (1962).
[20] Ralph, R. K., and H. G. Khorana, *J. Am. Chem. Soc.*, **83**, 2926 (1961).
[21] Ames, B. N., and D. T. Dubin, *J. Biol. Chem.*, **235**, 769 (1960).
[22] Chen, P. S., Jr., T. Y. Toribara, and H. Warner, *Anal. Chem.*, **28**, 1756 (1956).
[23] Heppel, L. A., D. R. Harkness, and R. J. Hilmoe, *J. Biol. Chem.*, **237**, 841 (1962).
[24] Razzell, W. E., and H. G. Khorana, *J. Biol. Chem.*, **234**, 2114 (1959).
[25] Kay, E. R. M., N. S. Simmons, and A. L. Dounce, *J. Am. Chem. Soc.*, **74**, 1724 (1952).
[26] Sherman, J. R., and J. Adler, *J. Biol. Chem.*, **238**, 873 (1962).
[27] Waley, S. G., and J. Watson, *Biochem. J.*, **55**, 328 (1953).
[28] Akabori, S., K. Ohno, and K. Narita, *Bull. Chem. Soc. Japan*, **25**, 214 (1952).
[29] Lowry, O. H., N. J. Rosebrough, A. L. Farr, and R. J. Randall, *J. Biol. Chem.*, **193**, 265 (1951).
[30] Nakamoto, T., and S. B. Weiss, these PROCEEDINGS, **48**, 880 (1962).
[31] Fox, C. F., W. S. Robinson, and S. B. Weiss, *Fed. Proc.*, **22**, 463 (1963).
[32] Krakow, J. S., and S. Ochoa, these PROCEEDINGS, **49**, 88 (1963).

THE INDEPENDENCE OF THE CONTINUUM HYPOTHESIS

BY PAUL J. COHEN*

DEPARTMENT OF MATHEMATICS, STANFORD UNIVERSITY

Communicated by Kurt Gödel, September 30, 1963

This is the first of two notes in which we outline a proof of the fact that the Continuum Hypothesis cannot be derived from the other axioms of set theory, including the Axiom of Choice. Since Gödel[1] has shown that the Continuum Hypothesis is consistent with these axioms, the independence of the hypothesis is thus established. We shall work with the usual axioms for Zermelo-Fraenkel set theory,[2] and by Z-F we shall denote these axioms without the Axiom of Choice, (but with the Axiom of Regularity). By a model for Z-F we shall always mean a collection of actual sets with the usual ϵ-relation satisfying Z-F. We use the standard definitions[2] for the set of integers ω, ordinal, and cardinal numbers.

THEOREM 1. *There are models for Z-F in which the following occur:*

(1) *There is a set* a, $a \subseteq \omega$ *such that* a *is not constructible in the sense of reference 3, yet the Axiom of Choice and the Generalized Continuum Hypothesis both hold.*

(2) *The continuum (i.e.,* $\mathcal{P}(\omega)$ *where* $\mathcal{P}$ *means power set) has no well-ordering.*

(3) *The Axiom of Choice holds, but* $\aleph_1 \neq 2^{\aleph_0}$.

(4) *The Axiom of Choice for countable pairs of elements in* $\mathcal{P}(\mathcal{P}(\omega))$ *fails.*

Only part 3 will be discussed in this paper. In parts 1 and 3 the universe is well-ordered by a single definable relation. Note that 4 implies that there is no simple ordering of $\mathcal{P}(\mathcal{P}(\omega))$. Since the Axiom of Constructibility implies the Generalized Continuum Hypothesis,[3] and the latter implies the Axiom of Choice,[5] Theorem 1 completely settles the question of the relative strength of these axioms.

Before giving details, we sketch the intuitive ideas involved. The starting point is the realization[1, 4] that no formula $\varphi(x)$ can be shown from the axioms of Z-F to have the property that the collection of all x satisfying it form a model for Z-F in which the Axiom of Constructibility ($V = L$,[3]) fails. Thus, to find such models, it seems natural to strengthen Z-F by postulating the existence of a set which is a

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。

変貌する「集合論」

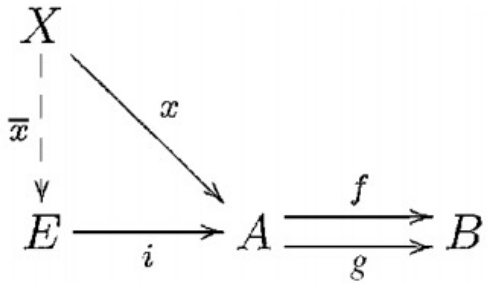
60年代のコーエンの仕事によって大きく射程を拡大した「集合論」だが、その後も大きな変化が続いている。

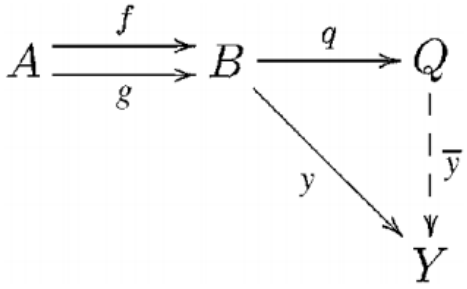
70年代には、LawvereのTopos Theoryが登場し、カテゴリー論の上に「集合論」は再構築され拡大した。代数幾何をはじめとする、他の数学の諸分野との関係も、一層密接なものとなった。

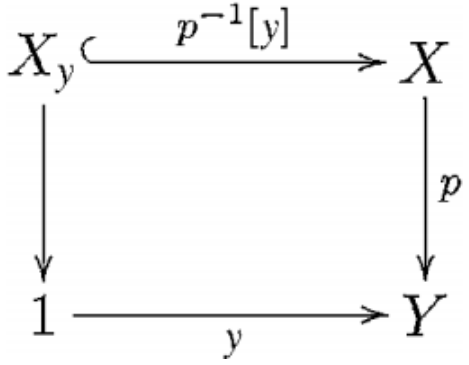
21世紀に入ってからの、VoevodskyのUnivalent Theory / Homotopy Type Theory の動きは重要である。惜しいことに、Voevodskyは、2017年に亡くなった。

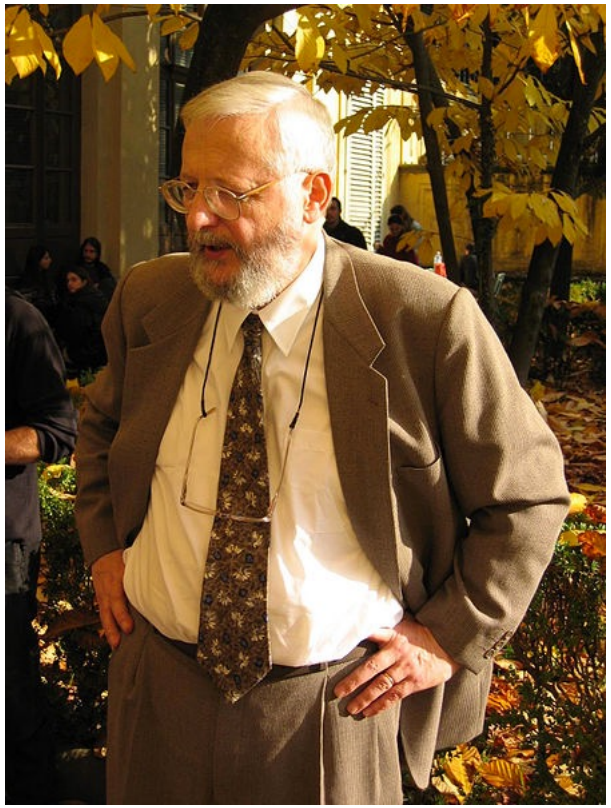
数学の基礎としての「集合論」は、時代とともに、そのスタイルを変えている。これらの動きは、数学の基礎について、新たな洞察を提供している。

Lawvere's Topos Theory









William Lawvere
1937-

Voevodsky's Univalent Theory

Vladimir Voevodsky
1966-2017

dim 0: Sets have elements/objects.

dim 1: Categories also have arrows between objects

dim 2: 2-categories have in addition arrows between those arrows.

⋮ ⋮

Identity between elements/objects

dim 0: standard equality between elements of a set.

dim 1: isomorphism between objects of a category

dim 2: equivalence between objects of a 2-category

⋮ ⋮

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.

連続体仮説の独立性証明の実行

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

blog「感慨 -- 思えば遠くに来たもんだ」から

https://maruyama097.blogspot.com/2022/03/blog-post_17.html

昨日、不思議な感動的な体験をしました。

月末のセミナーに向けて、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ようとしているのですが、どうしても取り上げたいトピックがありました。

一年ほど前、連続体仮説の独立性をコンピュータ上で証明できたという論文が出ていました。彼らは、その証明を GitHubで公開していました。「証明」はコンピュータで実行可能な「プログラム」の形をしているのです。

昨日、それを実行してみたのです。

僕のくたびれた非力なMacは、証明の準備にとりかかりました。メッセージをみると、証明に必要な情報をたくさん集めているのがわかります。一時間ほどかかって、ようやく準備ができましたようです。「証明して」とコマンドを打ち込んだら、15分ほどで、証明が終わりました！

不思議な気持ちになりました。

確かに、僕が命令して僕のマシンが働いたのですが、僕は「連続体仮説の独立性」を証明したのでしょうか？

「連続体仮説」というのは、数直線上の 0と1 の間に、何個の点があるのかという問題です。カントールはこの個数についてある仮説を立てたのですが、ついには狂気にいたるまでそれを証明することができませんでした。ヒルベルトは、この問題を 20世紀の数学が解くべき問題の筆頭に掲げました。

ゲーデルは、この仮説が集合論の他の公理とは矛盾しないことを示すことができました。この問題が最終的に解いたのは、コーエンです。彼は、連続体仮説がなりたつ集合論もあれば、それが成り立たない集合論もあることを示すことに成功します。それが「連続体仮説の独立性」証明です。

arXivもGitHubもない時代でした。僕がコーヘンの論文とその準備をしたゲーデルの無矛盾性証明のレクチャーノートを手に入れるのに、田舎にいたせいもあって、3ヶ月かかりました。PCもタブレットもないので、プリンストンの真っ赤な表紙のレクチャーノートをいつも持ち歩いて、ボロボロになりました。

カントール、ヒルベルト、ゲーデル、コーエンが苦闘した問題を、こたつの上の古いMacが、一時間ちょっとで解いたのです！　かつて「連続体仮説」フェチだった僕にとっては、Open-AI の「人工知能」が、数Ⅰレベルの「数学オリンピック」の問題のいくつかをといたことなんかより、はるかに大事件です。

もっとも、こうしたことは理論的には、頭では分かっていました。「コンピュータは、推論・証明する能力を持っている」でも、実感したのは初めてです。

ショートムービー「証明へのコンピュータの利用
-- 連続体仮説をコンピュータで解く」を公開しました。

https://youtu.be/Vq3wTUrbL5c?list=PLQIrJ0f9gMcPP8LOejaQQlYufAMEQbcdg

多分、たいした意味はないのですが、証明のLogも公開しました。

https://drive.google.com/file/d/1w-WDnFxqVBMDbvkS3AU_ay9c3wNsBABL/view?usp=sharing

Feit–Thompson定理の形式的証明


Georges Gonthier

マルレク「コンピュータと数学」 Part 3 コンピュータによる証明

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

2008年、GontierはAMSに次の論文を投稿する。

“Formal Proof—The FourColor Theorem”

https://www.ams.org/notices/200811/tx081101382p.pdf

現在、Gonthierらの証明は、GitHubで公開されている。

https://github.com/coq-community/fourcolor

```
maruyama@MacBook fourcolor % cd theories
maruyama@MacBook theories % ls
approx.v                job239to253.v           patch.v
birkhoff.v              job254to270.v           present.v
cfcolor.v               job271to278.v           present10.v
cfcontract.v            job279to282.v           present11.v
cfmap.v                 job283to286.v           present5.v
cfquiz.v                job287to290.v           present6.v
cfreducible.v           job291to294.v           present7.v
chromogram.v            job295to298.v           present8.v
color.v                 job299to302.v           present9.v
coloring.v              job303to306.v           quiz.v
combinatorial4ct.v      job307to310.v           quiztree.v
configurations.v        job311to314.v           real.glob
contract.v              job315to318.v           real.v
ctree.v                 job319to322.v           real.vo
ctreerestrict.v         job323to383.v           real.vok
cube.v                  job384to398.v           real.vos
dedekind.v              job399to438.v           realcategorical.v
discharge.v             job439to465.v           realplane.v
discretize.v            job466to485.v           realprop.glob
dyck.v                  job486to489.v           realprop.v
embed.v                 job490to494.v           realsyntax.glob
finitize.v              job495to498.v           realsyntax.v
```

```
finitize.v              job495to498.v           realsyntax.v
fourcolor.v             job499to502.v           realsyntax.vo
geometry.v              job503to506.v           realsyntax.vok
grid.v                  job507to510.v           realsyntax.vos
gridmap.v               job511to516.v           redpart.v
gtree.v                 job517to530.v           reducibility.v
gtreerestrict.v         job531to534.v           revsnip.v
hubcap.v                job535to541.v           sew.v
hypermap.v              job542to545.v           snip.v
initctree.v             job546to549.v           task001to214.v
initgtree.v             job550to553.v           task215to234.v
job001to106.v           job554to562.v           task235to282.v
job107to164.v           job563to588.v           task283to302.v
job165to189.v           job589to610.v           task303to322.v
job190to206.v           job611to617.v           task323to485.v
job207to214.v           job618to622.v           task486to506.v
job215to218.v           job623to633.v           task507to541.v
job219to222.v           jordan.v                task542to588.v
job223to226.v           kempe.v                 task589to633.v
job227to230.v           kempetree.v             unavoidability.v
job231to234.v           matte.v                 walkup.v
job235to238.v           part.v
maruyama@MacBook theories %
```

```
20
21  Section FourColorTheorem.
22
23  Variable Rmodel : Real.model.
24  Let R := Real.model_structure Rmodel.
25  Implicit Type m : map R.
26
27  Theorem four_color_finite m : finite_simple_map m -> colorable_with 4 m.
28  Proof.
29  intros fin_m.
30  pose proof (discretize.discretize_to_hypermap fin_m) as [G planarG colG].
31  exact (colG (combinatorial4ct.four_color_hypermap planarG)).
32  Qed.
33
34  Theorem four_color m : simple_map m -> colorable_with 4 m.
35  Proof. revert m; exact (finitize.compactness_extension four_color_finite). Qed.
36
37  End FourColorTheorem.
38
```

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

odd-order/AUTHORS

Andrea Asperti	University of Bologna - Microsoft Inria Joint Centre
Jeremy Avigad	Carnegie Mellon University - Microsoft Inria Joint Centre
Yves Bertot	Inria Sophia Antipolis - Microsoft Inria Joint Centre
Cyril Cohen	LIX École Polytechnique - Microsoft Inria Joint Centre
François Garillot	Microsoft Inria Joint Centre
Georges Gonthier	Microsoft Research Cambridge - Microsoft Inria Joint Centre
Stéphane Le Roux	Microsoft Inria Joint Centre
Assia Mahboubi	Inria Saclay - Microsoft Inria Joint Centre
Sidi Ould Biha	Inria Sophia Antipolis - Microsoft Inria Joint Centre
Ioana Pasca	Inria Sophia Antipolis - Microsoft Inria Joint Centre
Laurence Rideau	Inria Sophia Antipolis - Microsoft Inria Joint Centre
Alexey Solovyev	University of Pittsburgh
Enrico Tassi	Inria Saclay - Microsoft Inria Joint Centre
Laurent Théry	Inria Sophia A ntipolis - Microsoft Inria Joint Centre
Russell O'Connor	Mc Master University - Microsoft Inria Joint Centre

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
```

odd-order/theoriesの構成

```
maruyama@MacBook odd-order % cd theories
maruyama@MacBook theories % ls
BGappendixAB.v                  BGsection9.v
BGappendixC.v                   PFsection1.v
BGsection1.v                    PFsection10.v
BGsection10.v                   PFsection11.v
BGsection11.v                   PFsection12.v
BGsection12.v                   PFsection13.v
BGsection13.v                   PFsection14.v
BGsection14.v                   PFsection2.v
BGsection15.v                   PFsection3.v
BGsection16.v                   PFsection4.v
BGsection2.v                    PFsection5.v
BGsection3.v                    PFsection6.v
BGsection4.v                    PFsection7.v
BGsection5.v                    PFsection8.v
BGsection6.v                    PFsection9.v
BGsection7.v                    stripped_odd_order_theorem.v
BGsection8.v                    wielandt_fixpoint.v
maruyama@MacBook theories %
```

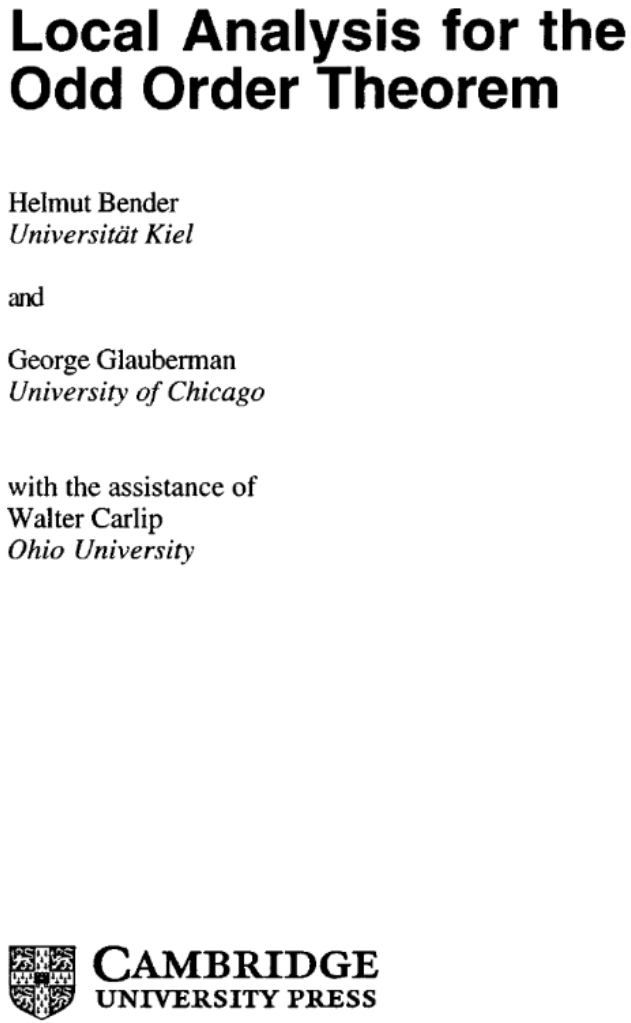
odd-order/theories/BGsection1.v

```
(*****************************************************************
************************)
(* This file contains most of the material in B & G, section 1,
including the *)
(* definitions:                                                  *)
```

odd-order/theories/PFsection1.v

```
(*****************************************************************
************************)
(* This file covers Peterfalvi, Section 1: Preliminary results.
*)
(*****************************************************************
************************)
```

(* This file contains most of the material in B & G, section 1,



Bender, Helmut;
Glauberman, George
(1994),

Local analysis for the odd order theorem,

London Mathematical Society Lecture Note Series, vol. 188, Cambridge University Press, ISBN 978-0-521-45716-3, MR 1311244

Contents

(* This file covers Peterfalvi, Section 1: Preliminary results.

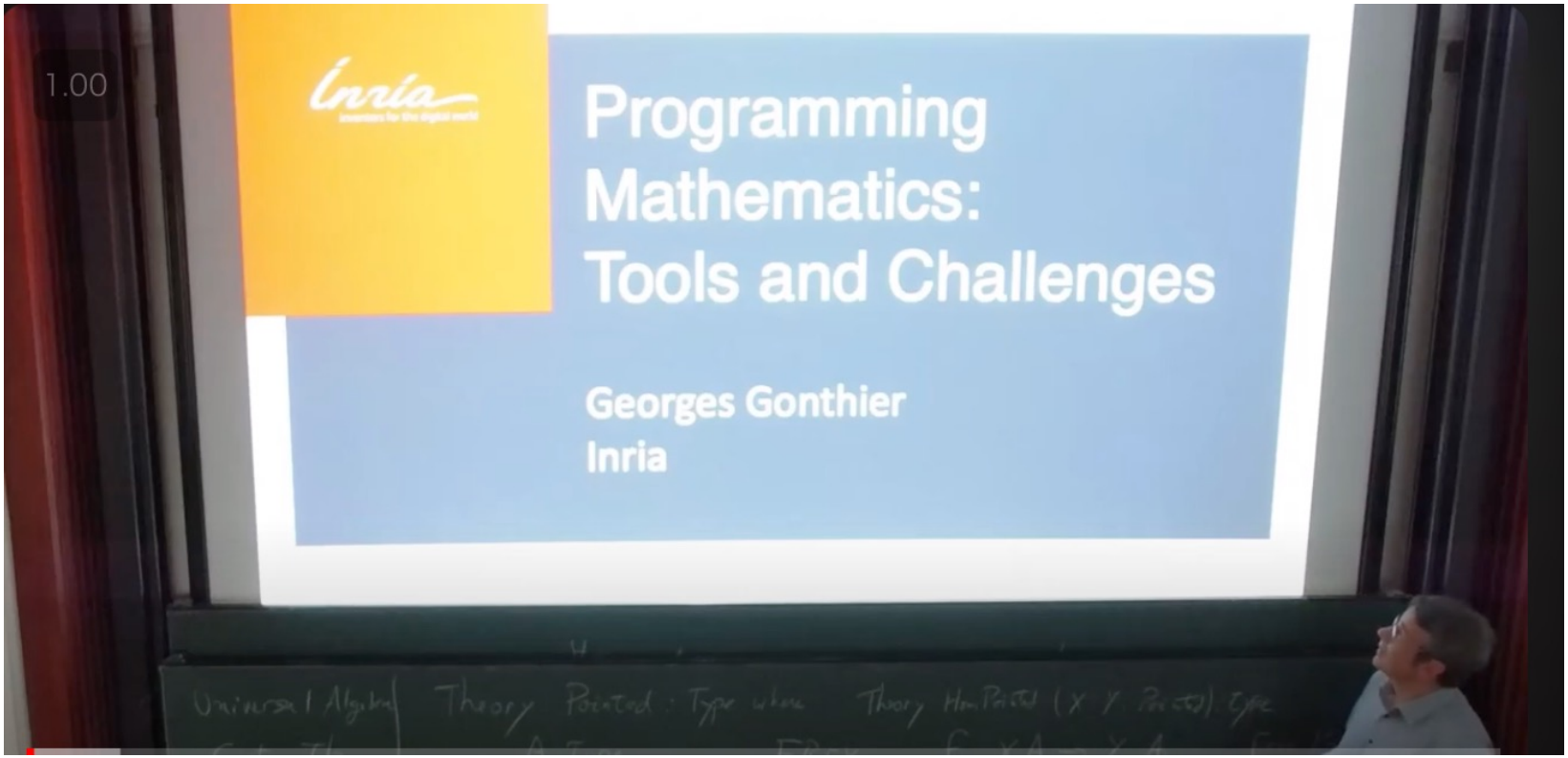
Peterfalvi, Thomas (2000),

Character theory for the odd order theorem,

London Mathematical Society Lecture Note Series, vol. 272,
Cambridge University Press,
doi:10.1017/CBO9780511565861,

ISBN 978-0-521-64660-4, MR 1747393

Georges Gonthier: Programming Mathematics: Tools and Challenges



2024/06/25
YouTube https://www.youtube.com/watch?v=fc5crdT3u_U

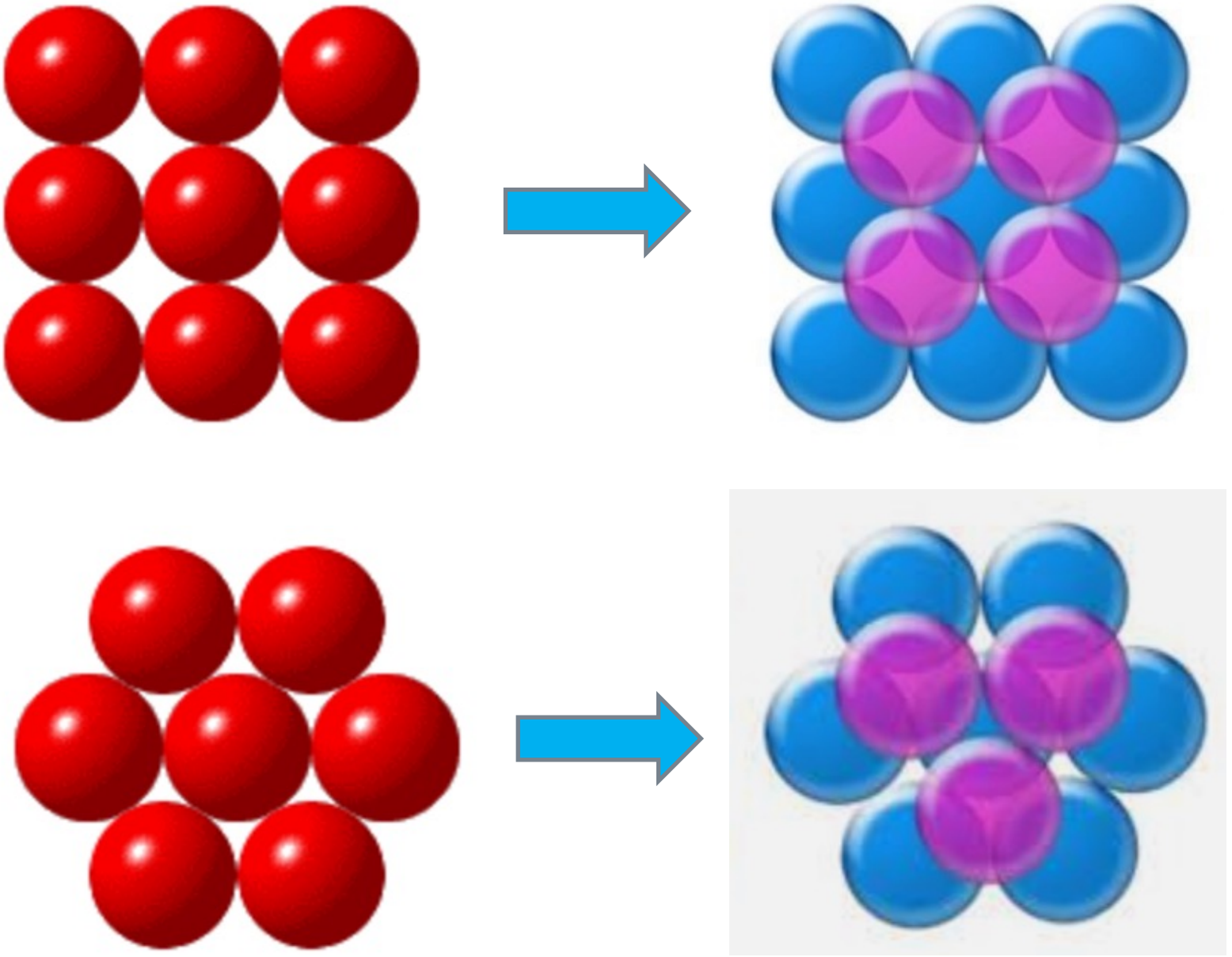
ケプラー予想の証明

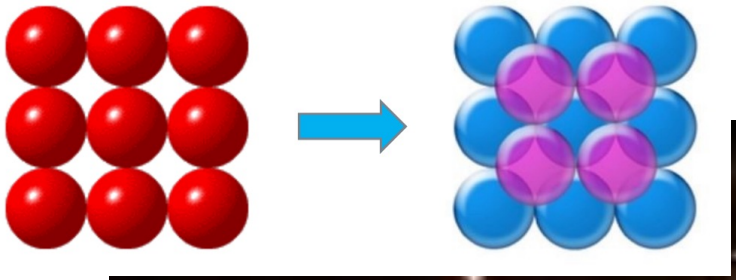
ケプラー予想

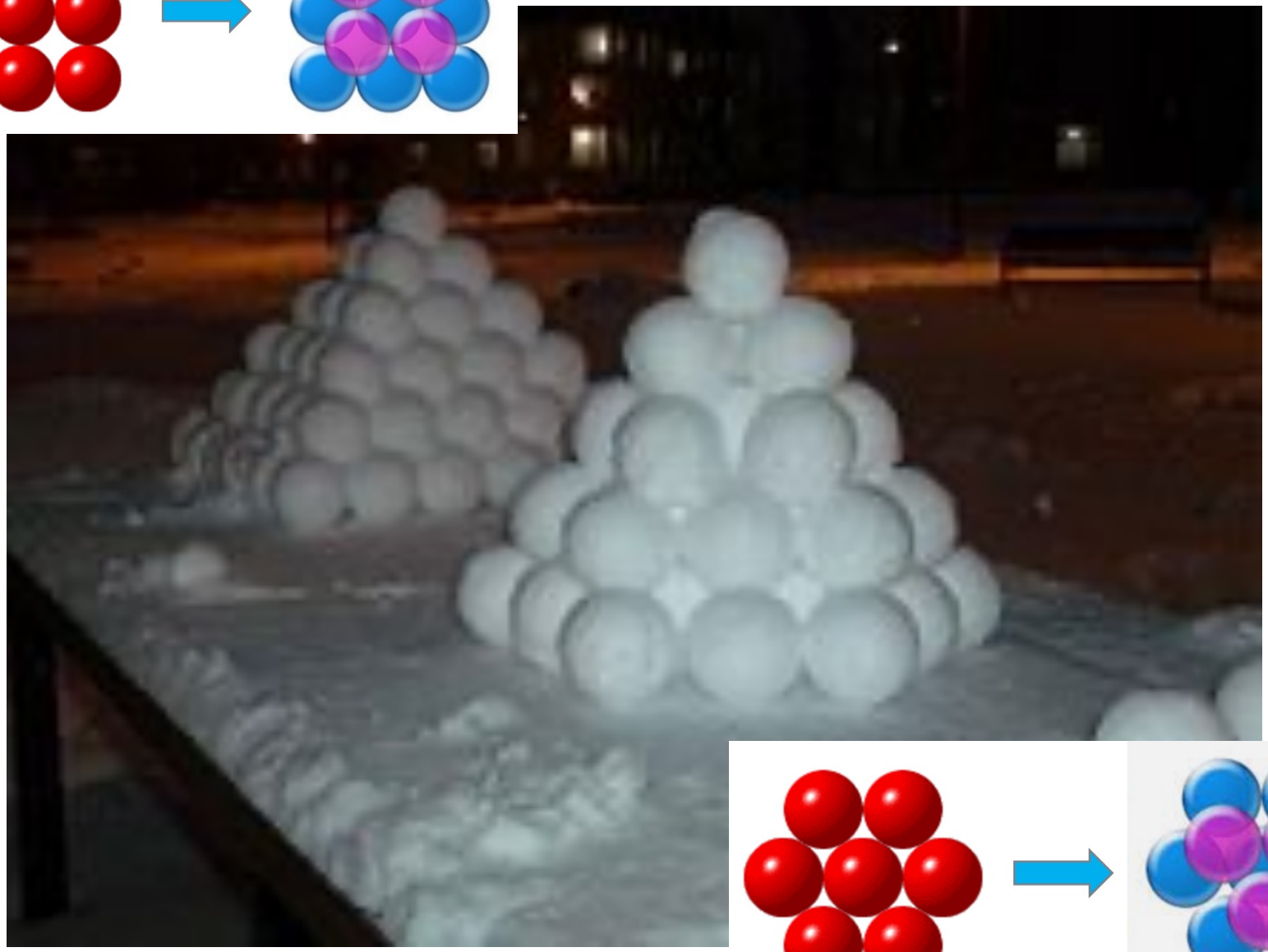
「ケプラー予想」は、入れ物の箱に同じ大きさのボールを、どのくらいたくさん詰め込むことができるかという問題に関する予想です。

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%程度になるといいます。箱の35%は、すきまになるということです。

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ました。







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、約74.05%で等しくなります。

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

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ました。これを「ケプラー予想」といiます。ケプラー自身は、この問題に証明を与えることはできませんでした。

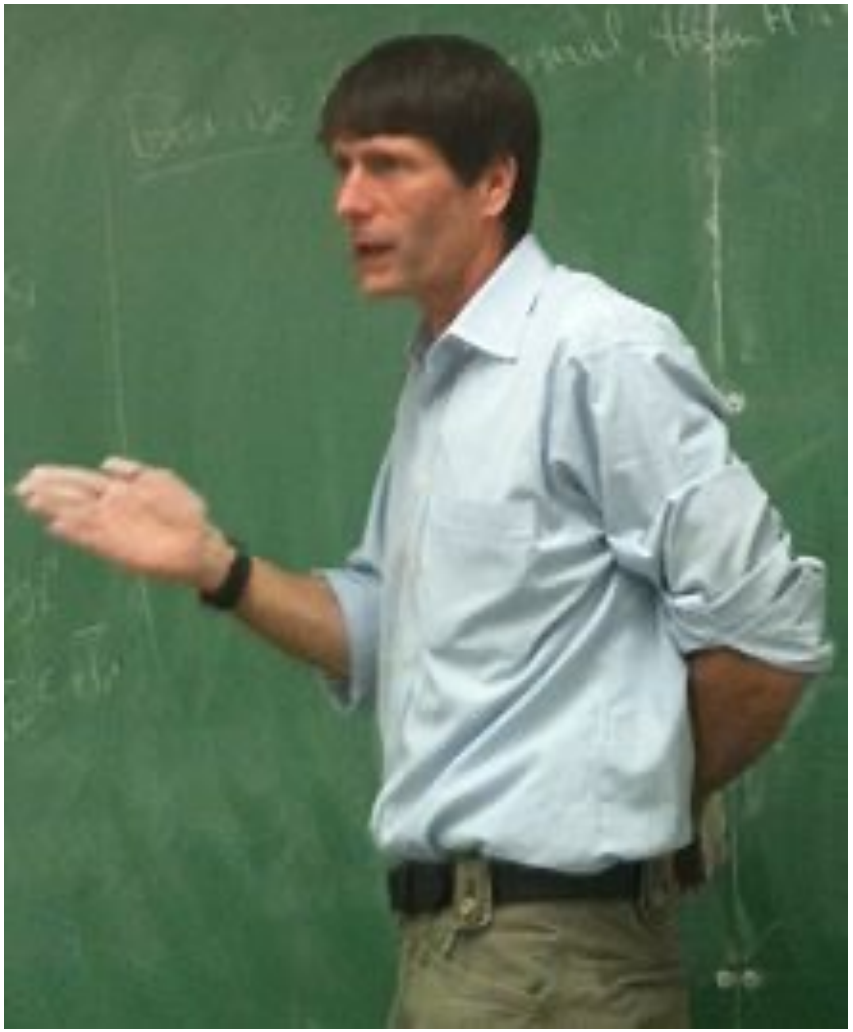
1900年のヒルベルトの20世紀の数学が解くべき23の問題の 18番目にこの問題は、取り上げられています。

ケプラー予想の400年

1611年

Johannes Kepler

2014年



Thomas Hales

Halesの論文、査読で留保される

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します。

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します。

ただ論文の査読者は「99%正しい」としたものの、1%を留保しました。

Hales ケプラー予想を解く

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ます。

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します。

2017年には、彼らの証明は、学会に受理されます。

Kepler conjecture 論文
https://arxiv.org/abs/1501.02155

[Submitted on 9 Jan 2015]

A formal proof of the Kepler conjecture

Thomas Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, Roland Zumkeller

This article describes a formal proof of the Kepler conjecture on dense sphere packings in a combination of the HOL Light and Isabelle proof assistants. This paper constitutes the official published account of the now completed Flyspeck project.

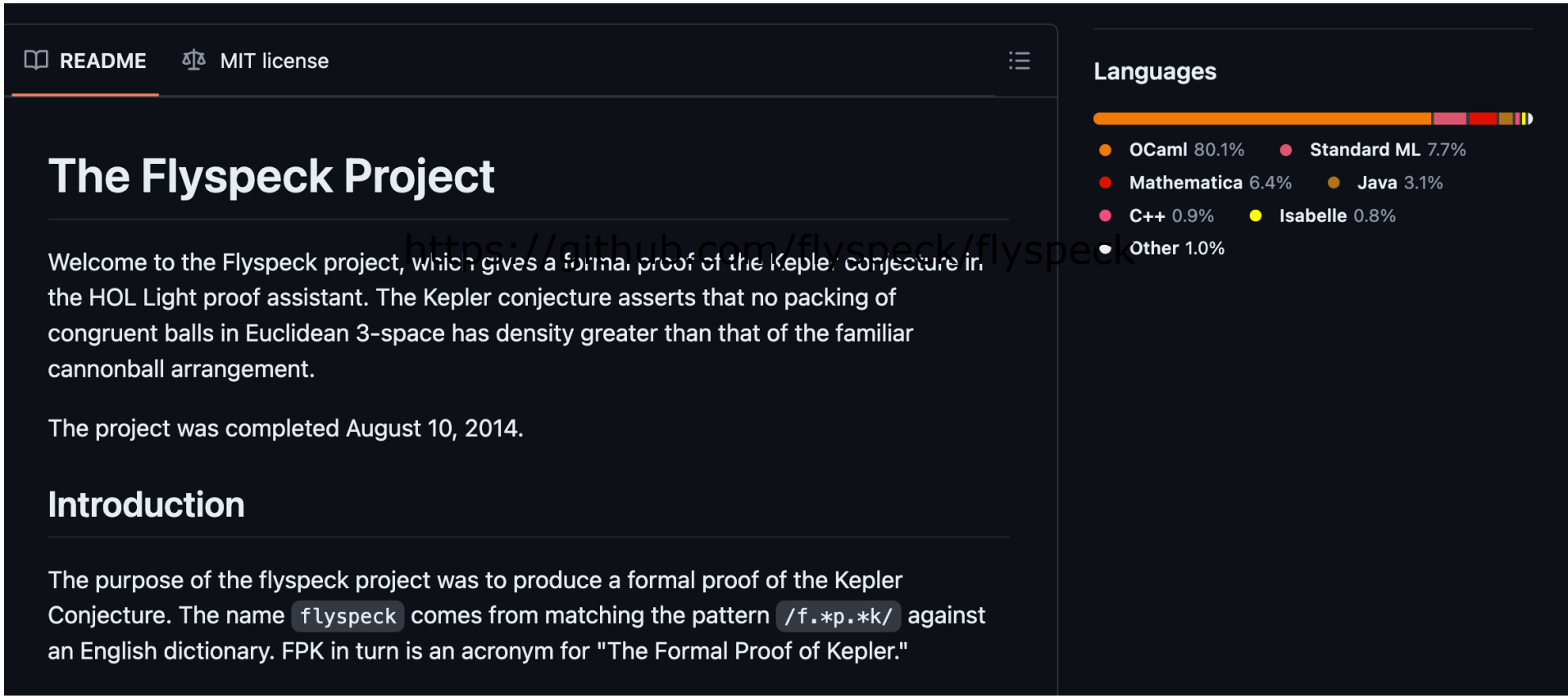
Comments: 21 pages
Subjects: **Metric Geometry (math.MG)**; Logic in Computer Science (cs.LO)
Cite as: arXiv:1501.02155 **[math.MG]**
(or arXiv:1501.02155v1 **[math.MG]** for this version)
https://doi.org/10.48550/arXiv.1501.02155

Submission history

From: Thomas Hales [view email]
[v1] Fri, 9 Jan 2015 14:32:24 UTC (33 KB)

Flyspeck Project GitHub

https://github.com/flyspeck/flyspeck



/flyspec/text_formalization/general/audit_formal_proof.hl

audit_formal_proof.hl ×

~/flyspeck/text_formalization/general/
audit_formal_proof.hl

audit_formal_proof.hl

HOL Light + Isabella

```
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
```

Grigory Perelmanのケース

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 ました。

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ありませんでした。

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年のことでした。

2006年、フィールズ賞の授与が決定されましたが、Perelmanはそれをを拒否します。

"[the prize] was completely irrelevant for me. Everybody understood that if the proof is correct, then no other recognition is needed."

"I'm not interested in money or fame, I don't want to be on display like an animal in a zoo"

2010年、Clay InstituteからMillennium Prizeとして、100万ドルの賞金が送られますが、Richard S. Hamiltonの業績が正しく評価されてないとして、その受け取りも拒否しました。

"the main reason is my disagreement with the organized mathematical community. I don't like their decisions, I consider them unjust."

彼は、今は数学の世界を離れているようです。