PROOF MINING

Lecture 1 - Hilbert's program and the rise of proof theory

Thomas Powell
University of Bath

Nordic Logic Summer School 2022

University of Bergen
13 June 2022

These slides are available at `https://t-powell.github.io/`.

Proof mining

Structure of course

HISTORICAL INTRODUCTION

TODAY: Hilbert's program and the rise of proof theory

IN-DEPTH EXPLORATION

TUESDAY: The functional interpretation of intuitionistic arithmetic.

WEDNESDAY: Classical logic and the negative translation.

HIGHLIGHTS

THURSDAY: Modern applications.

Outline

“Concrete” mathematics

Up to the 19th century, mathematics was primarily concerned with concrete objects which could be explicitly constructed. E.g.:

- Every number is either even or odd.
- There are infinitely many prime numbers.
- Every number can be written as the sum of four squares.
- Every continuous function can be integrated.
- Every non-constant polynomial over over the complex numbers has a root.

Proofs of these results yield concrete algorithms. E.g.:

- We can decide whether a number is even or odd.
- We can find the next prime number.
- For any number we can find four squares which sum to that number.
- We can compute the integral of f up to any desired accuracy.
- We can compute roots of polynomials up to any desired accuracy.

Infinitude of primes

Proposition

There are infinitely many prime numbers.

FUNDAMENTAL THEOREM OF ARITHMETIC: every number has a prime factorisation.

Proof (Aristotle, Euclid).

Suppose there are only finitely many primes and label them $p_1, \ldots, p_k$. Apply the fundamental theorem to $p_1 \cdot \cdots \cdot p_k + 1$ to find a prime factor. This cannot be any of the p_i. $\square$

What constructive information can we extract from this proof?

- The fundamental theorem of arithmetic gives us a factoring algorithm.
- Given primes $p_1, \ldots, p_k$, we simply factor $p_1 \cdots p_k + 1$ to find a new prime.

In other words, the proof comes equipped with an algorithm for finding the next prime.

We also derive a bound: for any number n there is a prime p with

$$n < p \leq n! + 1$$

Non-constructive mathematics

With the advent of modern mathematics in the 19th century, mathematicians started to reason about non-constructible objects.

- For every $f : \mathbb{N} \to \mathbb{N}$ there exists an n such that $f(n) \leq f(m)$ for all m.
- Every set of real numbers has a least upper bound.
- Every monotone, bounded sequence converges to a limit.
- Every ring has a maximal ideal.
- Every vector space has a basis.

In general, none of these existence results yield effective algorithms.

They give the existence of *ideal objects*, based purely on formal reasoning.

Question: Do these ideal objects really 'exist'?

Intuitively, we believe they do since we trust mathematical reasoning. But some rather bizarre phenomena may occur...

Example: irrational powers

Proposition

There are irrational numbers a, b such that a^b is rational.

Proof.

We know that $\sqrt{2}$ is irrational. What about $\sqrt{2}^{\sqrt{2}}$? We have two cases:

- If $\sqrt{2}^{\sqrt{2}}$ is rational, then set $a = b = \sqrt{2}$.
- Otherwise, set $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. We have,

$$a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$$

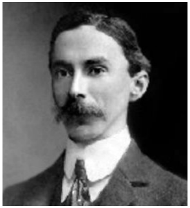
so a^b is rational as required. □

We have proved the proposition, but we do not know which of the cases hold!

Question: Is $\sqrt{2}^{\sqrt{2}}$ rational or irrational?! Answered by the Gelfond-Schneider theorem (it's irrational).

Down the rabbit hole...

Even worse, this style of mathematical reasoning can lead us down fallacious paths.



Example (Russell's paradox (B. Russell, 1901))

Let $R := \{x : x \notin x\}$. Is $R \in R$?

- If $R \in R$ then by definition we must have that $R \notin R$;
- But if $R \notin R$ then we must have $R \in R$.

We have a contradiction!

Conclusion: Our naive formulation of mathematics, in particular set theory, is *inconsistent*, and so cannot be trusted.

The foundational crisis

The discovery of various paradoxes (including Russell's paradox) led to the so-called foundational crisis at the turn of the century:

- Can we construct solid formal foundations for mathematics?
- Can we ensure that they are consistent, and do no succumb to paradoxes?
- In particular, can we give a formal treatment of set theory?

These led to the resurgence of fundamental philosophical questions:

- What *is* a mathematical proof?
- Can mathematics/arithmetic be reduced to pure logic?
- Do mathematical objects 'exist' in some abstract sense, or are they just symbols on a piece of paper?

Outline

Emerging philosophies

Two competing schools of thought emerged as a reaction to the foundational crisis:

INTUITIONISM (led by L. E. J. Brouwer, see [Iemhoff, 2016])

- Based on *semantics*.
- Mathematics is a mental construction: Something exists only if it can be constructed.
- Infinite sets, maximals ideals, limits are all dubious unless they can be built explicitly.

FORMALISM (led by D. Hilbert, see [Weir, 2015])

- Based on *syntax*.
- Mathematics is a game of symbols: something 'exists' if it can be derived from mathematical axioms by logical inference rules.
- Infinite sets, maximal ideals, limits are all fine, as long as our underlying logical system *can be trusted*.

We will follow the formalist approach, but take advantage of elegant ideas from both.

Hilbert's Program

In the early 20th century, Hilbert proposed a set of benchmarks for a satisfactory foundation of arithmetic:

Hilbert's Program, 1921

Find a collection of axioms and inference rules $\mathcal{P}$ for arithmetic which is:

1. **Complete:** all true statements in the language of arithmetic are provable in $\mathcal{P}$.
2. **Consistency:** $\mathcal{P}$ does not prove a contradiction.

Aside: But this statement is circular! To show that $\mathcal{P}$ is consistent we need to work in some other system, which in turn needs to be shown to be consistent etc. Hilbert was well aware of this, so he asked for the following further refinement:

(continued)

3. **Finitary consistency:** the fact that $\mathcal{P}$ is consistent is demonstrable using only simple *finitary* methods, whose validity *cannot be questioned*.

Unwinding Hilbert's formalism

The notion of 'finitary' in Hilbert's program is crucial and, indeed, a subject of debate. Hilbert asks us to distinguish object-level systems $\mathcal{P}$ from 'finitary' meta-level systems $\mathcal{N}$ where:

- $\mathcal{P}$ can reason about crazy objects which cannot be constructed (and which would be rejected by intuitionists).
- $\mathcal{N}$ is grounded in a world of numbers and simple arithmetic operations, which no reasonable person could doubt. It is unquestionably correct.

The meta-level system $\mathcal{N}$ should be adequate for proving the consistency of the object-level system $\mathcal{P}$. In particular:

$$\mathcal{N} \vdash \text{“}\mathcal{P}\text{ is consistent”} \tag{1}$$

Intuition: to trust $\mathcal{P}$, it is enough to trust $\mathcal{N}$.

Chasing dreams

Hilbert's program is a great idea! It is the gold standard of formalist philosophy, guaranteeing mathematical practice that is free from contradictions and paradoxes.

There is only one tiny catch...

... it doesn't work.

ENTER K. GÖDEL:

How Gödel impacts Hilbert

- We would assume our formal system $\mathcal{P}$ allows us, in particular, to reason about elementary arithmetic.
- **Gödel II:** Assume $\mathcal{P}$ is a consistent formal system which contains elementary arithmetic. Then
$$\mathcal{P} \nvdash \mathsf{Con}(\mathcal{P})$$
- Let $\mathcal{N}$ be our proposed finitary system. Then we would assume that $\mathcal{N}$ is a subsystem elementary arithmetic, and hence of $\mathcal{P}$. Therefore
$$\mathcal{N} \nvdash \mathsf{Con}(\mathcal{P})$$

The way around Gödel

In a very specific sense, Hilbert's program is impossible. But let's reconsider our assumption

> *... we would assume that $\mathcal{N}$ is a subsystem of elementary arithmetic ...*

Can we prove $\mathrm{Con}(\mathcal{P})$ in some system $\mathcal{N}$ which is

- not a subsystem of elementary arithmetic, but
- still 'finitary' in some sense?

What would be a good candidate for $\mathcal{N}$?

Revised Hilbert's program

A number of different approaches to Hilbert's program were proposed. In particular, Gentzen proposed that $\mathcal{N}$ could be a system of ordinals, and employed ordinal analysis - induction up to ε_0 - to prove the consistency of Peano arithmetic.

But there was another person who attempted to overcome the obstacle thrown down by Gödel...

... Gödel himself!

Outline

Reformulating Hilbert's program

Gödel's incompleteness theorems demonstrate that Hilbert's program could not be achieved, *but only in its most narrow sense*.

From a broader perspective, Hilbert's program was an extraordinary success, and continues in spirit in modern mathematics, particularly proof theory and theoretical computer science. Its legacy includes:

- The establishment of formalism: Proofs as syntactic objects which can be manipulated according to combinatorial rules.
- The connection between proofs and programs, and the development of powerful techniques which translate between the two.
- Introducing the idea that ordinary mathematical proofs can be studied as objects in their own right.

UNIVERSITY OF BATH

This course

This lecture course is about the success of Hilbert's program and the role it plays in research today.

We focus on **proof interpretations**: formal translations between strong logical theories $\mathcal{P}$ and 'finitary' systems $\mathcal{N}$

$$\mathcal{P} \mapsto \mathcal{N}$$

Our aims are to:

1. Introduce Goedel's Dialectica interpretation and explain how it works;
2. Emphasise the connection between proof and computation;
3. Explain how we can 'compute' fundamentally non-computable objects;
4. Give some examples of modern research that uses proof interpretations and ideas from Hilbert's program to obtain new results in mathematics.

For the remainder of this lecture we give an broad outline of the main ideas.

Proof Interpretations

Recall the original aim of Hilbert's program: Reduce a mathematical theory $\mathcal{P}$ to a very simple finitary theory $\mathcal{N}$, which we imagine as a map

$$\mathcal{P} \mapsto \mathcal{N}$$

The aim is that we would then have

$$\mathrm{Con}(\mathcal{N}) \Rightarrow \mathrm{Con}(\mathcal{P}).$$

In Hilbert's original formulation:

- $\mathcal{P}$ is a theory which contains a reasonable portion of 'ordinary mathematics', such as Peano arithmetic. We can use these theories to prove the existence of non-computable objects.
- $\mathcal{N}$ is a simple, finitistic theory. This was never fully specified by Hilbert, but the intuition was that it contains nothing beyond simple numbers and arithmetical operations, and is entirely computational.

Gödel's functional ('Dialectica') interpretation (1958)

One of the first responses to Hilbert's program was given by Gödel himself. His idea was to weaken the notion of 'finitary':

$$\underbrace{\text{PA}}_{\mathcal{P}} \mapsto \underbrace{\text{System }\mathsf{T}}_{\mathcal{N}}$$

He achieved one of the first *relative* consistency proofs of ordinary mathematics, namely:

$$\text{Con}(\mathsf{T}) \Rightarrow \text{Con}(\text{PA})$$

In Gödel's consistency proof:

- $\mathcal{P}$ is the theory of Peano arithmetic;
- $\mathcal{N}$ is a new theory called System T. It contains simple operations on numbers, but also more complicated things, such as recursion over higher types, which are not strictly finitary in Hilbert's sense.
- System T is essentially a simple functional programming language akin to Haskell.

Outline

UNIVERSITY OF BATH

Gödel's idea

Gödel's idea was to choose, as our 'finitary' system $\mathcal{N}$, a prototypical **programming language** called System T.

System T doesn't look like a programming language in the modern sense: Rather, it is a simple and clean mathematical system which allows us to define programs using **primitive recursion**.

For example, if we wanted to construct the factorial function we would simply write

$$f(0) = 1 \quad f(n+1) = (n+1) \cdot f(n)$$

So far everything looks very finitary indeed. The catch is that System T allows us to construct **functionals** of higher-type i.e. functions which take functions as arguments e.g.

$$F(g, n) = g(g(n+1)) + g(n-1)$$

In this sense, System T is a precursor to the modern **functional programming language**, which include Haskell and ML.

System T: The types

In functional languages, programs are usually assigned a **type**.

In System T, the types are defined by two simple rules:

- $\mathbb{N}$ is a type;
- If ρ and τ are types, so is $\rho \to \tau$.

So $\mathbb{N}$, $\mathbb{N} \to \mathbb{N}$, $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ etc. are all types. The latter is a **functional**: a function which takes a function as an input.

Key point: System T is not a subsystem of elementary arithmetic, since arithmetic also talks about numbers and functions...

System T: The terms

Objects - or terms - of System T are defined by the following rules. We write $t : \rho$ to denote 't has type ρ'.

- there are infinitely many variables $x^\rho, y^\rho, z^\rho, \ldots$ of each type
- $0 : \mathbb{N}$ is a term
- the successor function $s : \mathbb{N} \to \mathbb{N}$ is a term. We write $x + 1$ for $s(x)$.
- if $t : \rho \to \tau$ and $s : \rho$ are terms, then so is $t(s)$.
- if $t : \tau$ is a term and $x : \rho$ a variable, then $\lambda x.t : \rho \to \tau$ is a term.
- for each type, the recursor $R_\rho : \rho \to (\mathbb{N} \to \rho \to \rho) \to \mathbb{N} \to \rho$ is a term

These terms are governed by axioms, which characterise how they behave. In particular, the λ-operator allows us to construct functions from terms:

$$(\lambda x.t)(s) = t[s/x]$$

while the recursor allows us to carry out recursion:

$$R_\rho a f 0 = a$$
$$R_\rho a f (n+1) = f n (R_\rho a f n)$$

The Ackermann function in System T

Define $I : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N}$ by

$$\begin{aligned} I(g, 0) &= g(1) \\ I(g, n+1) &= g(I(g, n)) \end{aligned}$$

and then $A : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ by

$$\begin{aligned} A(0) &= \lambda n.n + 1 \\ A(m+1) &= I(A(m)) \end{aligned}$$

It now follows that

$$\begin{aligned} A(0, n) &= n + 1 \\ A(m+1, 0) &= I(A(m), 0) \\ &= A(m, 1) \\ A(m+1, n+1) &= I(A(m), n+1) \\ &= A(m, I(A(m), n)) \\ &= A(m, A(m+1, n)) \end{aligned}$$

System T: The rest

Formally speaking, System T is a logical system whose objects are the terms described just now, and whose axioms consist of those which govern the terms, together with axioms of propositional logic and a scheme of induction.

But for this course, **the details are not important!** Here's what you need to know:

- System T is a simple programming language, which allows us to construct and reason about simple recursive functions and functionals.
- It is not a subsystem of elementary arithmetic, because it talks about higher-order objects, but it is fundamentally computational, or 'finitary'.
- It is nevertheless surprisingly strong! We can not only define all primitive recursive functions, but also things like the Ackermann function.

Outline

G. Kreisel and the ‘unwinding of proofs’

To summarise:

- Gödel’s functional interpretation is to interpret a complicated logical theory in a simpler calculus of programs (i.e. programming language).
- Originally, this was to obtain relative consistency proofs - to reduce the ‘trustworthyness’ of ordinary mathematics to that of a simple programming language.

However, in the 1960s the Austrian logician G. Kreisel proposed using proof interpretations for a different purpose: to extract explicit computational information from existential statements, or more generally put:

> *“What more do we know if we have proved a theorem by restricted mean than if we merely know that it is true?”*

For example: If we have a concrete proof that an object x exists, can we analyse the proof to find some ‘computational information’ about x?

Computational information

What do we mean by ‘computational information’?

- A proof of $P \vee Q \rightsquigarrow$ a boolean which tells us which of P or Q one is true.
- A proof that a set $X \subseteq \mathbb{N}$ is infinite $\rightsquigarrow$ for each $n \in \mathbb{N}$ a way to find some $m > n$ with $m \in X$.
- A proof that there is a real number x satisfying $R(x) \rightsquigarrow$ a method for computing approximations to x up to any desired accuracy.

But what about those fundamentally non-constructive objects?

- A proof that a sequence converges $\rightsquigarrow$ A rate of convergence? Can we always find this?
- A proof that every vector space has a basis $\rightsquigarrow$???

This is a question central to the course.

Outline

The functional interpretation of intuitionistic arithmetic (Lecture 2)

Gödel's intuitionistic functional interpretation maps theorems A in Heyting arithmetic (i.e. Peano arithmetic but without the law of excluded-middle $P \vee \neg P$) to formulas of the form $\exists x \forall y A_D(x,y)$ where $A_D(x,y)$ is a quantifier-free formula of System T. The main soundness theorem states:

$$\text{If HA proves } A \text{ then System T proves } \forall y A_D(t,y)$$

where t is a term (i.e. a program) that can be formally extracted from the proof of A.

In **Lecture 2** we will:

1. discuss the notion of program extraction in more detail;
2. briefly outline the theory of Heyting arithmetic;
3. study the interpretation $A \mapsto A_D(x,y)$;
4. give an overview of the soundness theorem;
5. carry out some worked examples.

Classical logic and the negative translation (Lecture 3)

There are some things which cannot be explicitly constructed e.g. a function f witnessing the Halting problem.

$$f(e,x) := \begin{cases} 1 & \text{if } \{e\} \text{ terminates on input } x \\ 0 & \text{otherwise} \end{cases}$$

One can prove that such a function *exists*, but no computable f can be constructed.

So how does the functional interpretation treat this kind of thing?

In **Lecture 3** we will:

1. give some examples of existential statements which are fundamentally non-computable;
2. describe the kind of 'indirect' information we can nevertheless extract from the underlying classical proofs;
3. make this formal via the negative translation of classical logic into intuitionistic logic;
4. discuss the special case of $\forall\exists$ statements.

Proof interpretations today (Lecture 4)

The application of proof interpretations in *mathematics* and *computer science* has taken off in the last two decades, and is an active and exciting area of research today. We will outline several topics at the cutting edge, and describe a range of open questions.

Lecture 4 will provide a high level snapshot of present day research in applied proof theory...

Outline

References I

Iemhoff, R. (2016).
Intuitionism in the philosophy of mathematics.
In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 edition.

Weir, A. (2015).
Formalism in the philosophy of mathematics.
In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2015 edition.