HoTT seminar 2017

organization

- *Join us this semester to find out:*
- *whether it is true that "identity is equivalent to equivalence",*
- *why the homotopy groups of spheres and the algorithms for type checking are discussed in one and the same seminar*
- *if HoTT is the ultimate solution to the problem of formalizing mathematics in proof assistants.*
- *Homotopy Type Theory (HoTT) is a new approach to foundations of logic, programming and mathematics. It has an increasingly powerful impact on the development of the modern type-theory based tools for programming and verification, such as Coq proof assistant and Agda programming language…*

- Anybody from math?
- Anybody familiar with proof assistant/dependently typed programming languages?

- Our base: the free online HoTT book
  https://homotopytypetheory.org/book/

- The sources are available on GitHub:
  https://github.com/HoTT/book/

- And while we’re at it, if you’re familiar with Coq:
  https://github.com/HoTT/HoTT

- also good developments in Agda, Lean…

But we need to understand a lot beforehand

- intuitionistic vs. classical logic
- impredicate vs. predicative quantification
- intensional vs. extensional type theories
- propositional vs. judgemental/definitional equality (and identity types)

- all this preliminary material discussed in Ch. 1 and Appendix A
- at first sight, it might seem a deceptively easy reading
- It’s not. I don’t think it’s rational to assume we’ll get to Ch. 2 and beyond before June
- In fact, I think we should complement the reading of the opening chapter with some other material

Proposals

- Per Martin-Löf, *Intuitionistic Type Theory*, notes by G. Sambin of a series of lectures given in Padua 1980
http://intuitionistic.files.wordpress.com/2010/07/martin-lof-tt.pdf
and his other writings

- Chapter on identity in Adam Chlipala's CPDT book
http://adam.chlipala.net/cpdt/html/Cpdt.Equality.html

- Just to understand better Martin-Löf's notion of judgement:
opening pages of Frank Pfenning and Rowan Davies, *A Judgmental Reconstruction of Modal Logic*:
https://www.cs.cmu.edu/~fp/papers/mscs00.pdf

- Selected material from Morten Heine Sørensen, Pawel Urzyczyn, *Lectures on the Curry-Howard Isomorphism*, available via Science Direct on campus
http://www.sciencedirect.com/science/bookseries/0049237X/149

- Online entry in the Stanford Encyclopedia of Philosophy and references therein
https://plato.stanford.edu/entries/type-theory-intuitionistic/

- Selected slides from FOMUS 2016 (available from meeting's webpage, also via me)
http://fomus.weebly.com/talks-abstracts--videos.html

- Lectures of Nicola Gambino and others and SMC 2014 in Lyon:
http://smc2014.univ-lyon1.fr/doku.php?id=week1

- For more ambitious people, Thomas Streicher's habilitation
http://www.mathematik.tu-darmstadt.de/~streicher/HabilStreicher.pdf

Questions

- How many of you are likely to actively participate?
- Anybody not willing to receive a grade, but likely to give a talk?
- If you want to get a grade, **prepare an electronic presentation** (in exceptional cases handouts at least) and give us the file afterwards. You can base it on the HoTT book sources…
- … but if you're willing to fill one of early slots, you can get away with a purely blackboard presentation (though handouts would still be great)

some slides stolen from tomorrow’s intro to SemProg

(and also from Pierce, Zdancewic et al., UPenn)

Logic

- Logic is the field of study whose subject matter is *proofs*
- Volumes written about its central role in computer science
- Manna and Waldinger called it the calculus of computer science
- Halpern et al.'s paper *On the Unusual Effectiveness of Logic in Computer Science*
- In particular, the notion of inductive proofs ubiquitous in all of computer science.
    - You have surely seen them before (discrete math, analysis of algorithms …)
    - … but in this course we examine them more deeply

Tools for proofs

- Automated theorem provers (see FMSoft) provide “push-button” operation
  - given a proposition, return either *true*, *false*, or *ran out of time*
  - Although their capabilities limited to fairly specific sorts of reasoning, they have matured enough to be useful now in a huge variety of settings.
  - Examples of such tools include SAT solvers, SMT solvers, and model checkers.
- Proof assistants are hybrid tools
  - try to automate the more routine aspects of building proofs while depending on human guidance for more difficult aspects.
  - Examples: Isabelle, Agda, Lean, Twelf, ACL2, PVS, and Coq among many others.
- Why logic and type theory enter the picture?
- Logic in its earlier days went through similar labour pains as software science did later…

A: How do we know something is true?

B: We test it out

A: But that isn't truth; testing can only give us evidence.
How do we know something is **true**?

B: We prove it

A: How do we know that we have a proof?

B: We need to define what it means to be a proof.
A proof is a logical sequence of arguments, starting from some initial assumptions

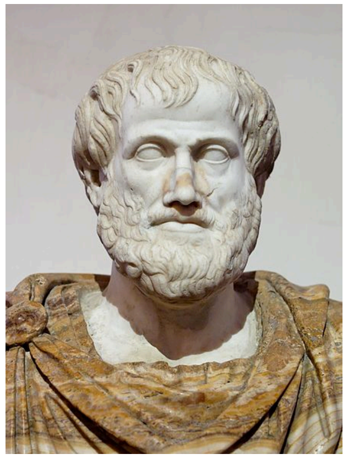
A: How do we know that we have a valid sequence of arguments? Can any list be a proof?

All humans are mortal

All Greeks are human

I am a Greek

B: No, no, no! We need to think about how we *think*….



Aristotle
384 – 322 BC

Euclid
~300 BC

(guest slide by Pierce, Zdancewic et al.)

First we need a language…

- Gottlob Frege: a German mathematician who started in geometry but became interested in logic and foundations of arithmetic.

- 1879 Published "*Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*" (Concept-Script: A Formal Language for Pure Thought Modeled on that of Arithmetic)
  - First rigorous treatment of functions and quantified variables
  - $\vdash A$, $\neg A$, $\forall x. F(x)$
  - First notation able to express arbitrarily complicated logical statements

Gottlob Frege
1848-1925

BEGRIFFSSCHRIFT,

EINE DER ARITHMETISCHEN NACHGEBILDETE

FORMELSPRACHE

DES REINEN DENKENS.

VON

Dr. GOTTLOB FREGE.

HALLE a/S.

VERLAG VON LOUIS NEBERT.

1879.

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)

Formalization of Arithmetic

- 1884: *Die Grundlagen der Arithmetik* (The Foundations of Arithmetic)
- 1893: *Grundgesetze der Arithmetik* (Basic Laws of Arithmetic, Vol. 1)
- 1903: *Grundgesetze der Arithmetik* (Basic Laws of Arithmetic, Vol. 2)
- Frege's Goals:
  - isolate logical principles of inference
  - derive laws of arithmetic from first principles
  - set mathematics on a solid foundation of logic

- David Hilbert: a German recognized as one of the most influential mathematicians ever.
  - algebra, axiomatization of geometry, physics,…
  - 1900: published his "23 Problems"
    - Problem #2: Prove
      are consistent

Hilbert
1943

The plot thickens…

Just as Volume 2 was going to print in 1903, Frege received a letter…

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)

Bertrand Russell

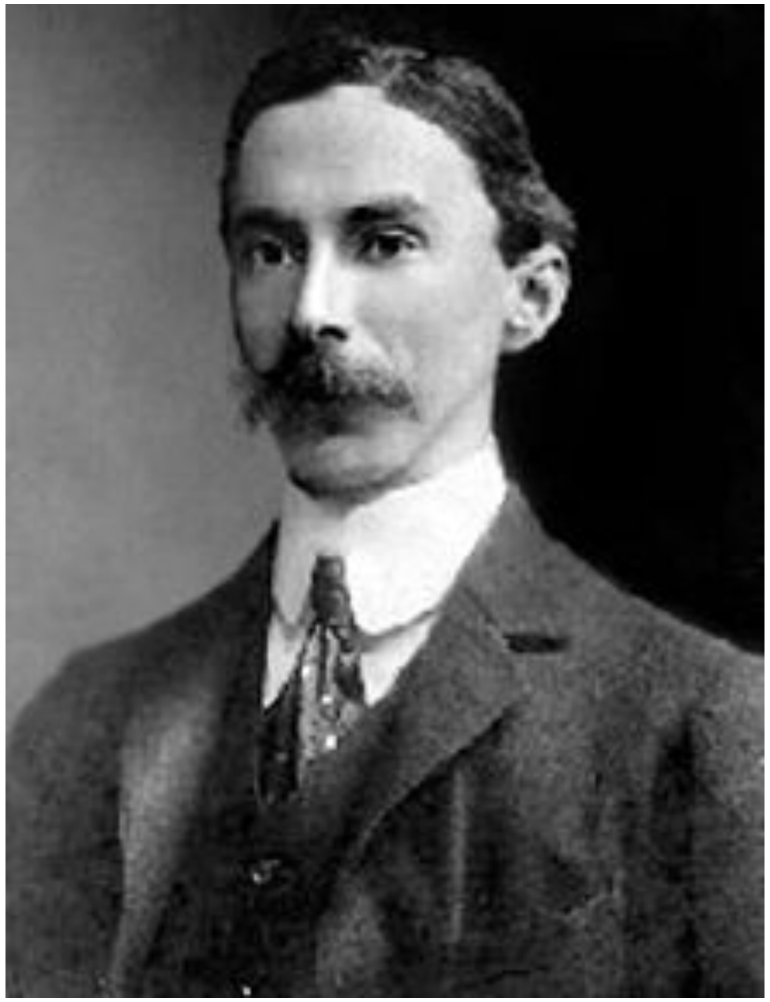
- *Russell's paradox:*

1. Set comprehension notation:
   $\{ x \mid P(x) \}$ "The set of x such that P(x)"

2. Let X be the set $\{ Y \mid Y \notin X \}$.

3. Ask the logical question:
   Does $X \in X$ hold?

4. Paradox! If $X \in X$ then $X \notin X$.
   If $X \notin X$ then $X \in X$.



Bertrand Russell
1872 - 1970

- Frege's language could derive Russell's paradox ⇒ it was *inconsistent*.
- Frege's logical system could derive anything. Oops(!!)

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)

Addendum to Frege's 1903 Book

"Hardly anything more unfortunate can befall a scientific writer than to have one of the foundations of his edifice shaken after the work is finished. This was the position I was placed in by a letter of Mr. Bertrand Russell, just when the printing of this volume was nearing its completion."

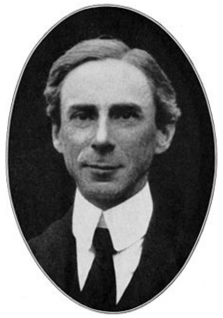
– Frege, 1903

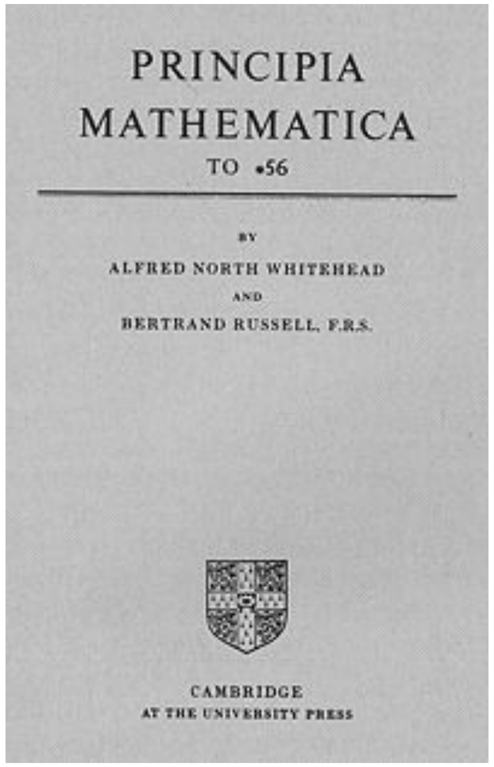
(guest slide by Pierce, Zdancewic et al.)

Aftermath of Frege and Russell

- Frege came up with a fix, but it made his logic trivial…
- 1908: Russell fixed the inconsistency of Frege's logic by developing a *theory of types*.
- 1910, 1912, 1913, (revised 1927): *Principia Mathematica* (Whitehead & Russell)
  - Goal: axioms and rules from which *all* mathematical truths could be derived.
  - It was a bit unwieldy…


Whitehead


Russell



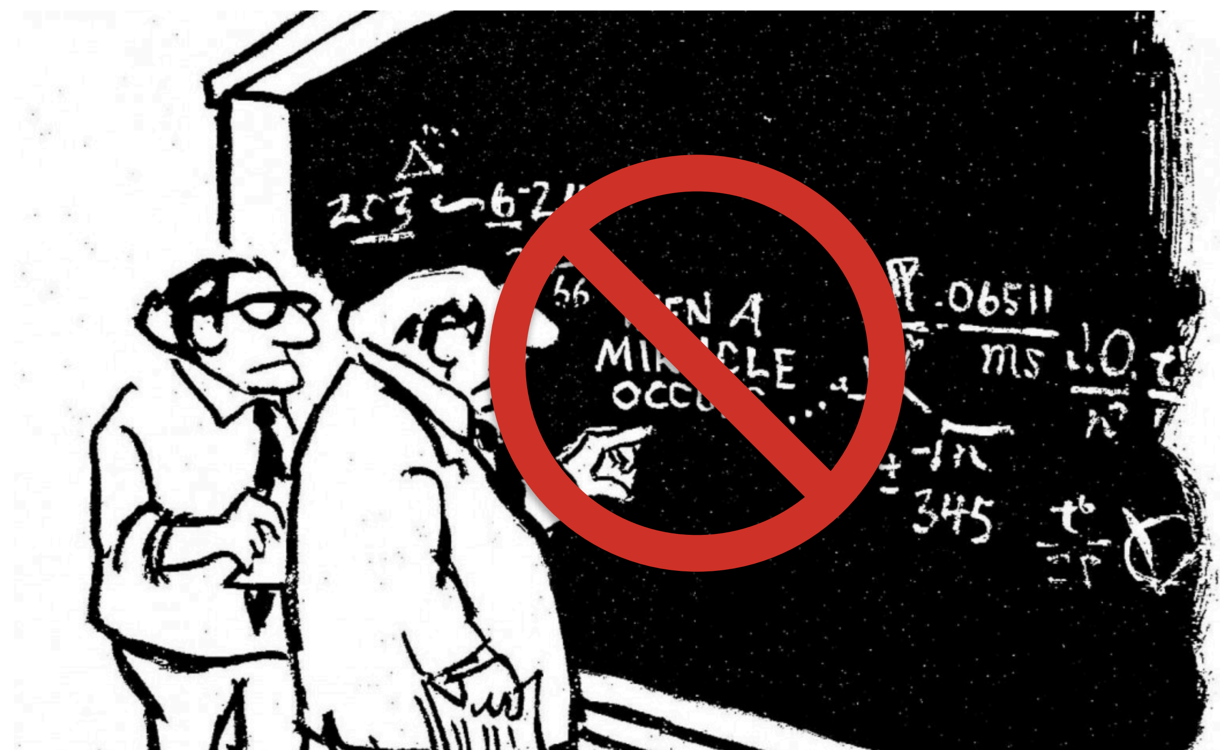
> "From this proposition it will follow,
> when arithmetical addition has been defined,
> that 1+1=2."
> —Volume I, 1st edition, *page 379*

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)

1920's: Hilbert's Program

A plan to secure the foundations of mathematics:

- Develop a *formal system* of all mathematics.
  - Mathematical statements should be written in a precise formal language
  - Mathematical proofs should proceed by well-specified rules
- Prove *completeness*
  - i.e. that all true mathematical statements can be proved
- Prove *consistency*
  - i.e. that no contradictory conclusions can be proved
- Prove *decidability*
  - i.e. there should be an algorithm for determining whether a given statement has a proof



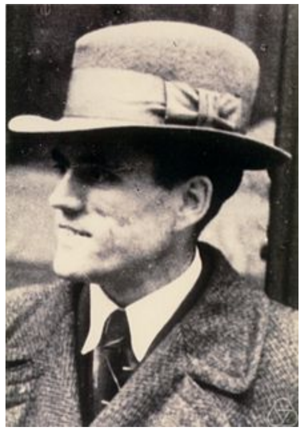
Things were going well, following Russell & Whitehead, until…

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)

Logic in the 1930s and 1940s

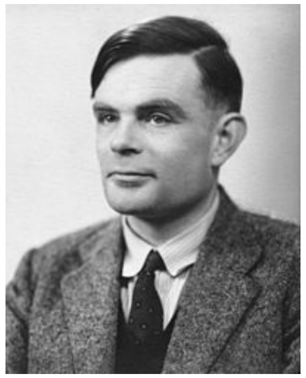
- 1931: Kurt Gödel's first and second incompleteness theorems.
  - Demonstrated that any consistent formal theory capable of expressing arithmetic cannot be complete.
  - Write down: "This statement is not provable." as an arithmetic statement.
- 1936: Genzen proves consistency of arithmetic.
- 1936: Church introduces the λ-calculus.
- 1936: Turing introduces Turing machines
  - Is there a decision procedure for arithmetic?
  - Answer: no it's undecidable
  - The famous "halting problem"
    - only in 1938 did Turing get his Ph.D.
- 1940: Church introduces the *simple theory of types*

Kurt Gödel
1906 - 1978



Gerhard Gentzen
1909 - 1945

Alonzo Church
1903 - 1995



Alan Turing
1912 - 1954

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)

Fast Forward…

- 1958 (Haskell Curry) and 1969 (William Howard) observe a remarkable correspondence:

Haskell Curry
1900 – 1982

William Howard
1926 –

types	~	propositions
programs	~	proofs
computation	~	simplification

N.G. de Bruijn
1918 - 2012

- 1967 – 1980's: N.G. de Bruijn runs Automath project
  - uses the Curry-Howard correspondence for computer-verified mathematics

Basis for modern type systems: OCaml, Haskell, Scala, Java, C#, …

- 1971: Jean-Yves Girard introduces System F
- 1972: Girard introduces Fω
- 1972: Per Marin-Löf introduces intuitionistic type theory
- 1974: John Reynolds independently discovers System F

(guest slide by Pierce, Zdancewic et al., with Wikipedia images)