The roots of computability theory

September 5, 2016

Algorithms

- An **algorithm** for a task or problem is a procedure that, if followed step by step and without any ingenuity, leads to the desired result/solution.
- "without ingenuity" means that in carrying out the algorithm, everything that we do is fully specified as part of the procedure.
- Feynman's three steps algorithm for solving any problem:

1. Write down the problem
2. Think very hard
3. Write down the solution

Not allowed:

requires ingenuity

Algorithms

- For an algorithm we have an **input** — that to which the algorithm is applied — and an **output** — that which the algorithm produces.

- Since its beginnings, the main business of mathematics has been to provide reliable algorithms to solve problems of many different kinds.

- Here are some examples.

Arithmetic

- Compute the **sum**, **product**, **quotient**, and **remainder** of two natural numbers.
  - input: $n, m \in \mathbb{N}$
  - output: $n+m$, $n \times m$, $n:m$, $rem(n,m)$

- Find the **greatest common divisor** of two natural numbers.
  - input: $n, m \in \mathbb{N}$
  - output: largest k which divides both n and m

Arithmetic

- **Decide whether a given natural number is prime.**
  - input: $n \in \mathbb{N}$
  - output: 'yes' or 'no'

- **Decompose a natural number n in prime factors.**
  - input: $n > 1$
  - output: prime numbers $p_1, \ldots, p_n$ and exponents $e_1, \ldots, e_n$ such that $n = p_1^{e_1} \times \ldots \times p_n^{e_n}$

Geometry

- **Construct an equilateral triangle** with sides of a given length.
  - input: a segment in the plane
  - output: an equilateral triangle with sides as long as the segment

- **Compute π** with any desired precision (Archimedes' algorithm).
  - input: a number n
  - output: the first n digits of the decimal expansion of π

Algebra, logic

- Resolve a **first degree equation** $ax+b=0$
  - input: rational numbers a,b
  - output: a rational number c such that $ac+b=0$

- **Decide whether a propositional formula φ is a tautology**
  - input: a propositional formula
  - output: 'yes' or 'no'

From algorithms in mathematics to a mathematics of algorithms

- As these examples show, algorithms are central to mathematics.
- Yet, it is only around 1930 that mathematicians turned attention to the algorithms themselves as mathematical objects.
- Before, the objects of the theories were numbers (arithmetic), points and lines (geometry), equations (algebra), formulas (logic), etc.
- In the 30s, various theories are developed in which the **algorithms themselves are the objects** whose properties are studied.

From algorithms in mathematics to a mathematics of algorithms

- These formal theories of algorithms made it possible to ask: which mathematical problems can be solved by means of algorithms? This is the subject of **computability theory**.
- And also: which problems can be solved by an efficient algorithm? This is the subject of **computational complexity theory**.
- This development was stimulated by **a number of important problems that were open at that time**. Let's look at some of these.

Problem 1: computable numbers

Problem 1: computable numbers

- Natural numbers are finite objects.
  They can be viewed as finite sequences of digits: 12828

- Real numbers are infinite objects.
  They can be viewed as infinite sequences of digits: $\pi = 3.141592\ldots$

- Problem: what does it mean to operate with a real number, given that one cannot specify all the digits anyway?

Problem 1: computable numbers

- Idea: for many real numbers $r \in \mathbb{R}$, there is an algorithm A_r that produces the digits of the decimal representation of r:
  - this is always the case for rationals: 1/3 = 0.3333 …
  - but also for the most salient irrational numbers: $\sqrt{2}$, $\sqrt{3}$, $\boldsymbol{\pi}$, **e**, …

- An algorithm is a **finite** specification of a procedure.
  We could take A_r to be a finite description of the real number r.

- Call a real number **computable** if its decimals can be produced by following an algorithm.

Problem 1: computable numbers

- Problem: **is every real number computable?**
- If so, this means that real numbers are finitely representable.
- To give a positive answer, we would just have to describe how to get an algorithm for any given number $r \in \mathbb{R}$.
- To give a negative answer, we need to show that for some number r, **there exists no algorithm to compute the digits of r**.
  This requires a mathematically precise notion of an algorithm.

Problem 2: deciding logical validity

Problem 2: deciding logical validity

An old dream: codify arguments in a language which is so transparent, and with inference rules so precise, that correctness of reasoning is just a matter of mathematical correctness:

"Obviously, once this is performed, every paralogism is nothing but a calculation mistake […] If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice for them to take their pencils in their hands and to sit down at the abacus, and say to each other (and if they so wish also to a friend called to help): Let us calculate [*calculemus*]."

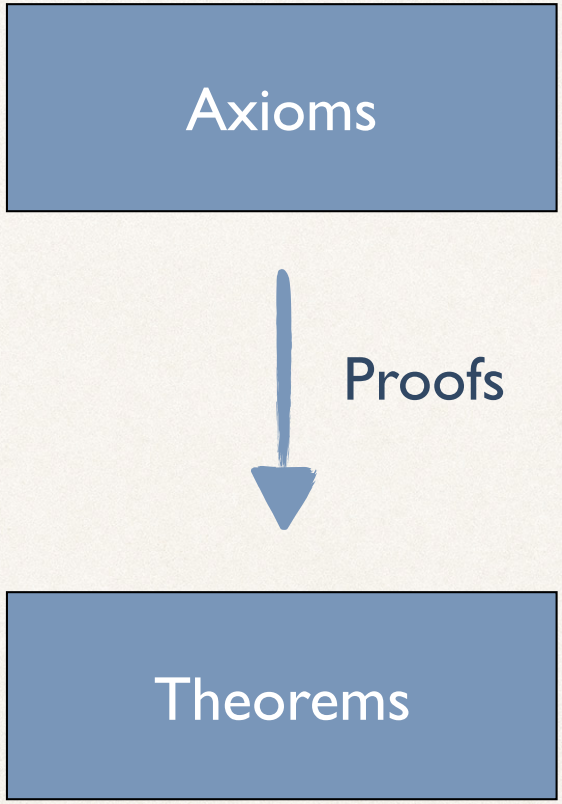
Leibniz, *De arte characteristica ad perficiendas scientias ratione nitentes*

Problem 2: deciding logical validity

In part, Leibniz's dream became a reality with modern formal logic:

- sentences of a language and their meaning are defined mathematically;
- the relation of consequence is defined in mathematical terms;
- whether a statement follows from a set of premises is therefore a mathematical question;
- moreover, for standard first-order logic, a complete set of inference rules exists: consequences can be proved by using these simple rules.

Problem 2: deciding logical validity



Problem 2: deciding logical validity

- Now, consider an axiomatic theory formulated in first-order logic. If we wish to know whether something follows from the theory, can we really say "calculemus"?

- Precisely: **is there an algorithm that allows us to decide whether a given first-order sentence is a consequence of a set of axioms?**

- This problem was posed by David Hilbert in 1928 and is known as the **decision problem** (aka "entscheidungsproblem").

- Again, to give a positive answer we just need to describe an algorithm, but for a negative answer we need to exclude that an algorithm exists, which requires a precise mathematical notion of algorithm.

Problem 2: deciding logical validity

- The decision problem has a **crucial importance for mathematics**: virtually all of mathematics (arithmetic, calculus, geometry, algebra, …) can be encoded in the axiomatic theory of sets (ZFC).

- If the answer to the decision problem is positive, then given any mathematical conjecture, we could formalize it and **decide in a mechanical way whether it is a theorem or not**.

- This means that we could check the validity of a conjecture in the same way as we check the correctness of an addition.

Problem 3: axiomatizing mathematics

Problem 3: axiomatizing mathematics

- In the beginning of the 20th century, there was much turmoil concerning the foundations of mathematics.
- Geometry and calculus had been reduced to the theory of sets.
- But what are sets? How can we know which sets exist?
  How about, e.g., alleged sets which have no finite description?
- Mathematicians disagreed about these questions and what principles were acceptable in a proof. Contradictions were discovered.
- But how could mathematics be certain if its basic notions are so unclear and controversial?

Problem 3: axiomatizing mathematics

- Hilbert's solution: mathematics is not concerned with what objects are, but only with what properties they have.

  "One must be able to say at all times—instead of points, straight lines, and planes—tables, chairs, and beer mugs"

- Mathematics proceeds by postulating the truth of some statements (axioms), and deriving the truth of other statements (theorems).

- Thus, we never manipulate infinite objects (sets, real numbers, etc.) but only statements about them, which are finite objects.

Problem 3: axiomatizing mathematics

With Hilbert's perspective, certainty is recovered:

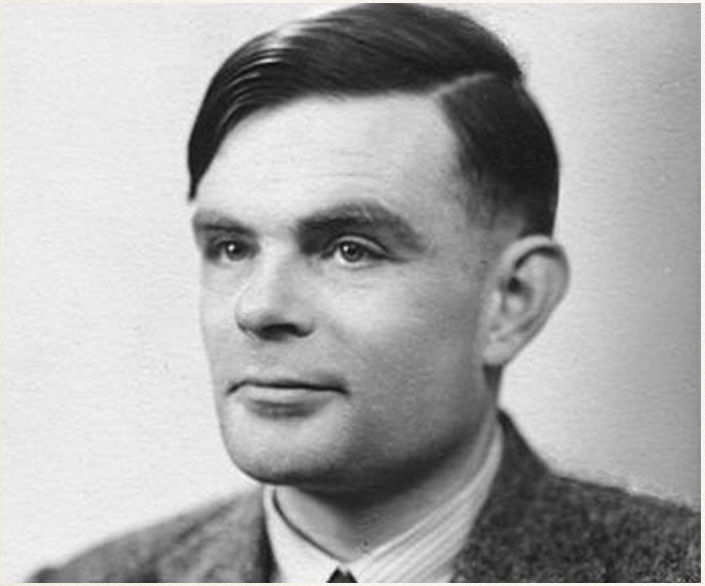
- we must be able to decide whether something is an axiom [the axiomatization is *effective*]
- we know what counts as a correct deduction from axioms;
- so, there is no room for disagreement over whether something really counts as a theorem or not.

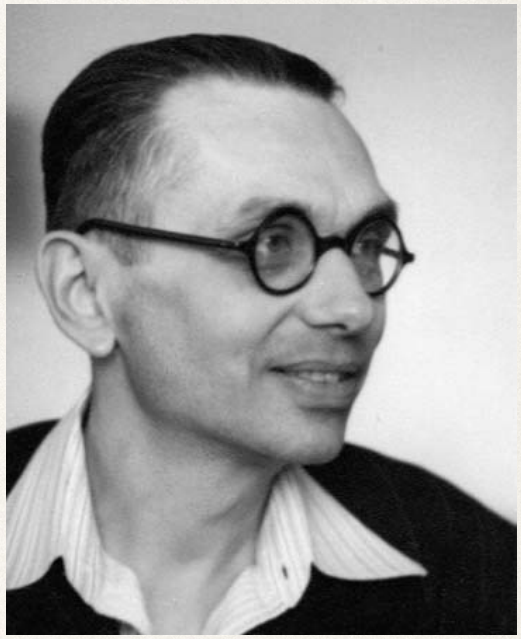
Problem 3: axiomatizing mathematics

- General problem: can we make our axiomatization rich enough to capture all the properties of the structure that we aim to characterize?
- More concretely, consider a language in which we can express all elementary statements of arithmetic, concerning $+$, $\times$, $:$, and so on.
- **Completeness problem**: can we give an axiomatization of arithmetic from which all true statements on numbers can be proved?
- This **axiomatization should be effective**, that is, there should be a procedure to tell whether something counts as an axiom or not. [Otherwise, we could trivially take anything that is true as an axiom.]
- To answer our question, we need to know what counts as a procedure.

Computability theory

The pioneers of computability





Alan Turing (1912-54)	Alonzo Church (1902-95)	Kurt Gödel (1906-78)
Cambridge, Manchester	Princeton, UCLA	Vienna, Princeton
Turing machines	Lambda calculus	Recursive functions

Computability theory answers our questions

1. **Computable numbers problem**: are all real numbers computable?

   No. For some $r \in \mathbb{R}$, there is no algorithm to compute their decimals.
   (Turing, 1936)

2. **Decision problem**: give a decision procedure to determine whether a sentence of first-order logic follows from a set of premises.

   No such procedure exists. (Church's theorem: Church, 1936, Turing, 1936)

3. **Completeness problem**: provide a complete and effective axiomatization of arithmetic (and mathematics more generally).

   No such axiomatization exists. Any effective axiomatization of arithmetic is incomplete, i.e., leaves some statements undecided.
   (Incompleteness theorem: Gödel, 1931)

Computability theory answers our questions

- The work on computability didn't just settle deep theoretical questions.
- It also provided explicit mathematical models of computation.
- These models inspired the development of physical computers. (VonNeumann architecture, 1945, regarded as first modern computer)
- And laid the foundations of new disciplines: computer science and AI.

Structure of the course

Classes	Topic	Main question
1-4	Models of computation	How can we analyze computation?
5-8	Introduction to computability	What can be computed?
9-10	Consequences for logic and mathematics	Can theoremhood be decided? Can maths be fully axiomatized?
11-12	Introduction to complexity	What can be computed efficiently?

Why is this relevant for artificial intelligence?

1. Work on computability is an amazing feat of AI (*ante litteram*): for the first time, a human cognitive ability (computing) was mathematically analyzed and replicated on a machine.

2. Computability theory sets mathematical limits to what a machine can do, which are independent of technological advances.

3. Gödel's theorem is a deep and mysterious results about reasoning: it shows that reasoning within a system of rules always has blind spots, which become visible by reasoning about the rules themselves.

4. Computational complexity considerations play an important role in designing and assessing all kinds of algorithms.

That’s all for today!