THE LIMITS OF PROOF

Finnur Lárusson

University of Adelaide

2008

Truth and provability

The job of the mathematician is to discover new truths about mathematical objects and their relationships. Such truths are established by proving them.

Truth and provability

The job of the mathematician is to discover new truths about mathematical objects and their relationships. Such truths are established by proving them.

Fundamental question. Can every mathematical truth be proved (by a sufficiently clever being) or are there truths that will forever lie beyond the reach of proof?

Truth and provability

The job of the mathematician is to discover new truths about mathematical objects and their relationships. Such truths are established by proving them.

Fundamental question. Can every mathematical truth be proved (by a sufficiently clever being) or are there truths that will forever lie beyond the reach of proof?

Mathematics can be turned on itself to investigate this question. The branch of mathematics that studies mathematical arguments is called *mathematical logic*.

Truth and provability

The job of the mathematician is to discover new truths about mathematical objects and their relationships. Such truths are established by proving them.

Fundamental question. Can every mathematical truth be proved (by a sufficiently clever being) or are there truths that will forever lie beyond the reach of proof?

Mathematics can be turned on itself to investigate this question. The branch of mathematics that studies mathematical arguments is called *mathematical logic*.

In this talk, we will see that under certain assumptions about proofs, there are truths that cannot be proved. You must decide for yourself whether you think these assumptions are valid!

First attempt: a countability argument

A proof is a finite piece of text. So the set of proofs is countable: just list them alphabetically.

First attempt: a countability argument

A proof is a finite piece of text. So the set of proofs is countable: just list them alphabetically.

The set of subsets of $\mathbb{N}$ is not countable: no list contains them all.

First attempt: a countability argument

A proof is a finite piece of text. So the set of proofs is countable: just list them alphabetically.

The set of subsets of $\mathbb{N}$ is not countable: no list contains them all.

Namely, if $A_1, A_2, A_3, \ldots$ is a list of subsets of $\mathbb{N}$, then the set

$$\{n \in \mathbb{N} : n \notin A_n\}$$

is not on the list (Cantor's diagonal argument).

First attempt: a countability argument

A proof is a finite piece of text. So the set of proofs is countable: just list them alphabetically.

The set of subsets of $\mathbb{N}$ is not countable: no list contains them all.

Namely, if $A_1, A_2, A_3, \ldots$ is a list of subsets of $\mathbb{N}$, then the set

$$\{n \in \mathbb{N} : n \notin A_n\}$$

is not on the list (Cantor's diagonal argument).

So there are uncountably many truths, such as

$$n \in X$$

for every subset X of $\mathbb{N}$ and every $n \in X$.

We need a better approach

Problem 1. We can sometimes prove infinitely many truths in a single proof. Think of induction!

We need a better approach

Problem 1. We can sometimes prove infinitely many truths in a single proof. Think of induction!

Problem 2. Even if someone could convincingly argue that there is a subset X of $\mathbb{N}$ and $n \in X$ such that the true statement $n \in X$ cannot be proved, we would like an explicit example of such X.

We need a better approach

Problem 1. We can sometimes prove infinitely many truths in a single proof. Think of induction!

Problem 2. Even if someone could convincingly argue that there is a subset X of $\mathbb{N}$ and $n \in X$ such that the true statement $n \in X$ cannot be proved, we would like an explicit example of such X.

We will define a particular subset B of $\mathbb{N}$ and show that for some $n \in B$ it cannot be proved that $n \in B$ — under certain commonly-made assumptions about proofs.

Programs

A *program* (algorithm, Turing machine, mechanical procedure, ...)
• can be written in any programming language
• has finite length
• runs on a physical computer not subject to malfunction or limitations of memory or time

Programs

A *program* (algorithm, Turing machine, mechanical procedure, ...)
- can be written in any programming language
- has finite length
- runs on a physical computer not subject to malfunction or limitations of memory or time

We can list all programs in a sequence $P_1, P_2, P_3, \ldots,$ for example in alphabetical order.

Programs

A *program* (algorithm, Turing machine, mechanical procedure, ...)
• can be written in any programming language
• has finite length
• runs on a physical computer not subject to malfunction or limitations of memory or time

We can list all programs in a sequence $P_1, P_2, P_3, \ldots,$ for example in alphabetical order.

We can take the input of a program to be a natural number.
As it runs, the program may, from time to time, output a natural number.
Depending on the input, the program halts after finitely many steps, or runs forever.

The Halting Problem

Does P_n halt on input m?

The Halting Problem

Does P_n halt on input m?

In 1936, Alan Turing showed that the Halting Problem is not algorithmically solvable.

The Halting Problem

Does P_n halt on input m?

In 1936, Alan Turing showed that the Halting Problem is not algorithmically solvable.

Proof. (Cantor again!) Suppose there was a program Q such that

$$Q(n) = \begin{cases} 1 & \text{if } P_n \text{ halts on input } n, \\ 0 & \text{otherwise.} \end{cases}$$

The Halting Problem

Does P_n halt on input m?

In 1936, Alan Turing showed that the Halting Problem is not algorithmically solvable.

Proof. (Cantor again!) Suppose there was a program Q such that

$$Q(n) = \begin{cases} 1 & \text{if } P_n \text{ halts on input } n, \\ 0 & \text{otherwise.} \end{cases}$$

Define a program P as follows:

$$P(n) = \begin{cases} P_n(n) + 1 & \text{if } Q(n) = 1, \\ 0 & \text{if } Q(n) = 0. \end{cases}$$

The Halting Problem

Does P_n halt on input m?

In 1936, Alan Turing showed that the Halting Problem is not algorithmically solvable.

Proof. (Cantor again!) Suppose there was a program Q such that

$$Q(n) = \begin{cases} 1 & \text{if } P_n \text{ halts on input } n, \\ 0 & \text{otherwise.} \end{cases}$$

Define a program P as follows:

$$P(n) = \begin{cases} P_n(n) + 1 & \text{if } Q(n) = 1, \\ 0 & \text{if } Q(n) = 0. \end{cases}$$

Then P is not on the list $P_1, P_2, P_3, \ldots$. $\square$

A remarkable subset of $\mathbb{N}$

Let

$$A = \{n \in \mathbb{N} : P_n \text{ halts on input } n\}.$$

A remarkable subset of $\mathbb{N}$

Let

$$A = \{n \in \mathbb{N} : P_n \text{ halts on input } n\}.$$

By Turing's result, A is not *computable (recursive)*: there is no program Q such that

$$Q(n) = \begin{cases} 1 & \text{if } n \in A, \\ 0 & \text{if } n \notin A. \end{cases}$$

A remarkable subset of $\mathbb{N}$

Let

$$A = \{n \in \mathbb{N} : P_n \text{ halts on input } n\}.$$

By Turing's result, A is not *computable (recursive)*: there is no program Q such that

$$Q(n) = \begin{cases} 1 & \text{if } n \in A, \\ 0 & \text{if } n \notin A. \end{cases}$$

However, A is *listable (recursively enumerable)*: there is a program that spits out all the numbers in A (in some order) and no others.

A remarkable subset of $\mathbb{N}$

Let

$$A = \{n \in \mathbb{N} : P_n \text{ halts on input } n\}.$$

By Turing's result, A is not *computable (recursive)*: there is no program Q such that

$$Q(n) = \begin{cases} 1 & \text{if } n \in A, \\ 0 & \text{if } n \notin A. \end{cases}$$

However, A is *listable (recursively enumerable)*: there is a program that spits out all the numbers in A (in some order) and no others.

It follows that $B = \mathbb{N} \setminus A$ is not listable.

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .

. . . we assume that **proofs are like programs**.

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .
. . . we assume that **proofs are like programs**.

Namely, suppose that we can codify our axioms and rules of inference in a formal language so that a computer can generate all possible proofs, one after another, . . .

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .
. . . we assume that **proofs are like programs**.

Namely, suppose that we can codify our axioms and rules of inference in a formal language so that a computer can generate all possible proofs, one after another, . . .
. . . in such a way that only true statements have proofs.

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .
. . . we assume that **proofs are like programs**.

Namely, suppose that we can codify our axioms and rules of inference in a formal language so that a computer can generate all possible proofs, one after another, . . .
. . . in such a way that only true statements have proofs.

Then there is $n \in B$ such that it is not provable that $n \in B$.

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .
. . . we assume that **proofs are like programs**.

Namely, suppose that we can codify our axioms and rules of inference in a formal language so that a computer can generate all possible proofs, one after another, . . .
. . . in such a way that only true statements have proofs.

Then there is $n \in B$ such that it is not provable that $n \in B$.

Otherwise, we could mechanically list B as follows:
Look through all proofs, one after another, and each time you find a proof, for some $n \in \mathbb{N}$, that $n \in B$, output n.

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if . . .
. . . we assume that **proofs are like programs**.

Namely, suppose that we can codify our axioms and rules of inference in a formal language so that a computer can generate all possible proofs, one after another, . . .
. . . in such a way that only true statements have proofs.

Then there is $n \in B$ such that it is not provable that $n \in B$.

Otherwise, we could mechanically list B as follows:
Look through all proofs, one after another, and each time you find a proof, for some $n \in \mathbb{N}$, that $n \in B$, output n.

But B is not listable!

A truth that cannot be proved

We can now exhibit a true statement that cannot be proved if ...
... we assume that **proofs are like programs**.

Namely, suppose that we can codify our axioms and rules of inference in a formal language so that a computer can generate all possible proofs, one after another, ...
... in such a way that only true statements have proofs.

Then there is $n \in B$ such that it is not provable that $n \in B$.

Otherwise, we could mechanically list B as follows:
Look through all proofs, one after another, and each time you find a proof, for some $n \in \mathbb{N}$, that $n \in B$, output n.

But B is not listable!

This proves *Gödel's First Incompleteness Theorem (1931)*.

Hilbert's 10th problem

Amazing fact. There is a polynomial $P(x_0, \ldots, x_m)$ with integer coefficients such that

$$A = \{n \in \mathbb{N} : \exists\ x_1, \ldots, x_m \in \mathbb{N} \text{ s.t. } P(n, x_1, \ldots, x_m) = 0\}.$$

Hilbert's 10th problem

Amazing fact. There is a polynomial $P(x_0, \dots, x_m)$ with integer coefficients such that

$$A = \{n \in \mathbb{N} : \exists\ x_1, \dots, x_m \in \mathbb{N} \text{ s.t. } P(n, x_1, \dots, x_m) = 0\}.$$

DPRM Theorem (1949–1970). A subset of $\mathbb{N}$ is listable if and only if it is *diophantine* (given by a polynomial as above).

Hilbert's 10th problem

Amazing fact. There is a polynomial $P(x_0, \ldots, x_m)$ with integer coefficients such that

$$A = \{n \in \mathbb{N} : \exists\ x_1, \ldots, x_m \in \mathbb{N} \text{ s.t. } P(n, x_1, \ldots, x_m) = 0\}.$$

DPRM Theorem (1949–1970). A subset of $\mathbb{N}$ is listable if and only if it is *diophantine* (given by a polynomial as above).

Since A is not computable, there is no algorithm that takes $n \in \mathbb{N}$ as input and tells us whether or not the equation $P(n, x_1, \ldots, x_m) = 0$ can be solved with $x_1, \ldots, x_m \in \mathbb{N}$.

Hilbert's 10th problem

Amazing fact. There is a polynomial $P(x_0, \ldots, x_m)$ with integer coefficients such that

$$A = \{n \in \mathbb{N} : \exists\ x_1, \ldots, x_m \in \mathbb{N} \text{ s.t. } P(n, x_1, \ldots, x_m) = 0\}.$$

DPRM Theorem (1949–1970). A subset of $\mathbb{N}$ is listable if and only if it is *diophantine* (given by a polynomial as above).

Since A is not computable, there is no algorithm that takes $n \in \mathbb{N}$ as input and tells us whether or not the equation $P(n, x_1, \ldots, x_m) = 0$ can be solved with $x_1, \ldots, x_m \in \mathbb{N}$.

This gives a negative solution to Hilbert's 10th problem.

Should we be depressed . . .

. . . about these results?

Should we be depressed . . .

. . . about these results?

Not at all. They show that mathematics is even more interesting and more challenging than we might have expected!

Should we be depressed . . .

. . . about these results?

Not at all. They show that mathematics is even more interesting and more challenging than we might have expected!

References for further reading.

Robert S. Wolf. *A tour through mathematical logic.* The Mathematical Association of America, 2005.

Martin Davis. *The incompleteness theorem.* Notices of the American Mathematical Society **53** (2006), no. 4, 414–418.

Bjorn Poonen. *Undecidability in number theory.* Notices of the American Mathematical Society **55** (2008), no. 3, 344–350.

Notices of the American Mathematical Society are freely available at `www.ams.org`