15-251: Great Theoretical Ideas in Computer Science
Lecture 22
November 13, 2014

Gödel’s Incompleteness Theorems

Proving the famous
“Gödel Incompleteness Theorem”
is **easy** if you use computer science.

It’s a Great Application of Theoretical Computer Science to mathematics.

It’s so easy, let’s kill some time reviewing older material.

15-251: Great Theoretical Ideas in Computer Science
Fall 2014, Lecture 3

Axiomatic Systems & Logic

$$\frac{P,\ P \rightarrow Q}{Q}$$

First-Order Logic:

stuff like $\forall x\ (\neg(x=\mathbf{r})\rightarrow \mathrm{IsSmarter}(\mathrm{Father}(\mathbf{r}),\mathrm{Father}(x)))$.

Given a vocabulary, some FOL sentences are “**valid**”:
i.e., “true for all possible interpretations”,
“automatically true, for ‘purely logical’ reasons”.

e.g.:

$(\forall x(x=\mathbf{a}))\rightarrow(\mathrm{Next}(\mathbf{a})=\mathbf{a})$

$\forall x\ \forall y\ ((x=\mathbf{a}\wedge y=\mathbf{b})\rightarrow(\mathrm{Func}(x,y)=\mathrm{Func}(\mathbf{a},\mathbf{b})))$

$\mathrm{IsCool}(\mathbf{c})\rightarrow(\exists x\ \mathrm{IsCool}(x))$

Gödel’s **Completeness** Theorem (1929):

“There’s a (computable) axiomatic system for validity.”

The “LOGIC TEXTBOOK” axiomatic system has:

a bunch of axioms (all of which are obviously valid sentences);

one deduction rule: from A and A→B, deduce B.

Every “theorem” in this system is valid (i.e., system is sound)

Gödel showed: *every valid sentence is a theorem*

(i.e., the system is complete)

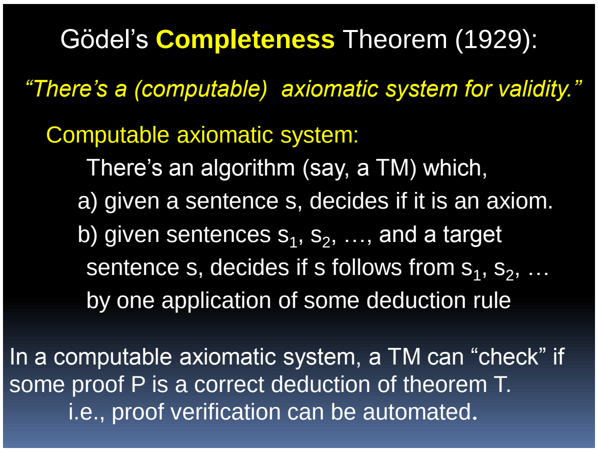
Gödel’s **Completeness** Theorem (1929):

“There’s a (computable) axiomatic system for validity.”

Actually, LOGIC TEXTBOOK does not have finitely many axioms. It has finitely many “axiom schema”. For example…

“if A is any sentence, then $A\vee\neg A$ is an axiom”

“if IsR is any relation-name and c is any constant-name, then $\mathrm{IsR}(\mathbf{c})\rightarrow(\exists x\ \mathrm{IsR}(x))$ is an axiom”



Gödel's **Completeness** Theorem (1929):

"There's a (computable) axiomatic system for validity."

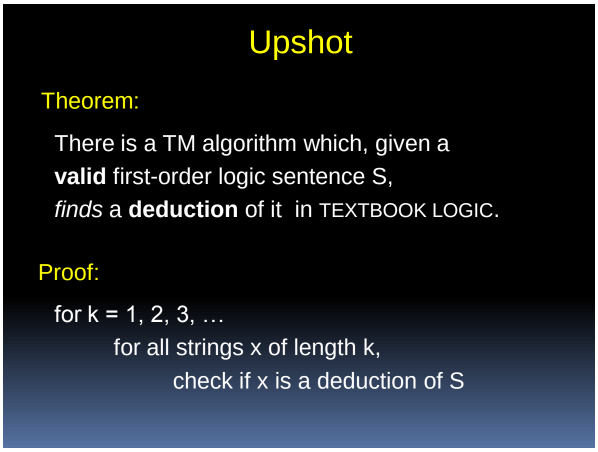
Computable axiomatic system:

There's an algorithm (say, a TM) which,

a) given a sentence s, decides if it is an axiom.

b) given sentences s_1, s_2, …, and a target sentence s, decides if s follows from s_1, s_2, … by one application of some deduction rule

In a computable axiomatic system, a TM can "check" if some proof P is a correct deduction of theorem T. i.e., proof verification can be automated.



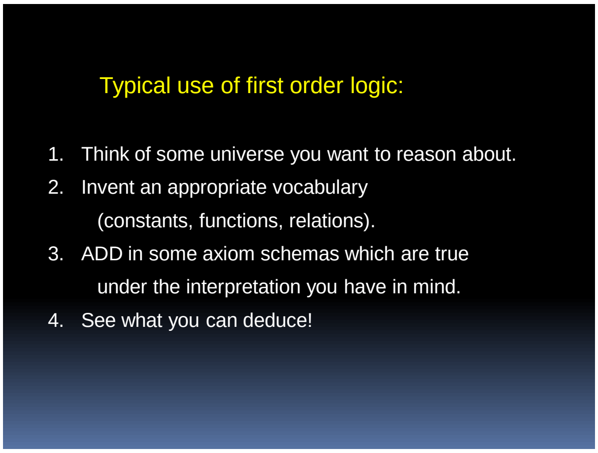
Upshot

Theorem:

There is a TM algorithm which, given a **valid** first-order logic sentence S, *finds* a **deduction** of it in TEXTBOOK LOGIC.

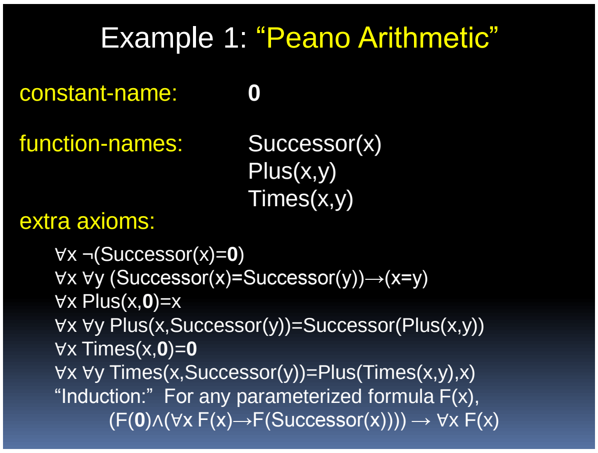
Proof:

```
for k = 1, 2, 3, …
    for all strings x of length k,
        check if x is a deduction of S
```



Typical use of first order logic:

1. Think of some universe you want to reason about.
2. Invent an appropriate vocabulary
   (constants, functions, relations).
3. ADD in some axiom schemas which are true
   under the interpretation you have in mind.
4. See what you can deduce!



Example 1: "Peano Arithmetic"

constant-name: **0**

function-names: Successor(x)
Plus(x,y)
Times(x,y)

extra axioms:

$\forall x\ \neg(\text{Successor}(x)=\mathbf{0})$

$\forall x\ \forall y\ (\text{Successor}(x)=\text{Successor}(y))\rightarrow(x=y)$

$\forall x\ \text{Plus}(x,\mathbf{0})=x$

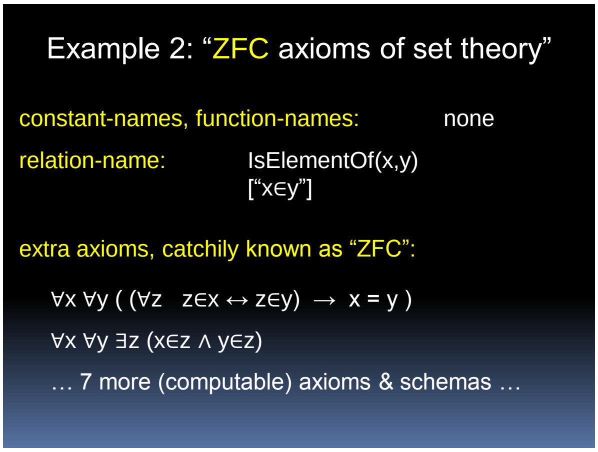
$\forall x\ \forall y\ \text{Plus}(x,\text{Successor}(y))=\text{Successor}(\text{Plus}(x,y))$

$\forall x\ \text{Times}(x,\mathbf{0})=\mathbf{0}$

$\forall x\ \forall y\ \text{Times}(x,\text{Successor}(y))=\text{Plus}(\text{Times}(x,y),x)$

"Induction:" For any parameterized formula F(x),

$(F(\mathbf{0})\wedge(\forall x\ F(x)\rightarrow F(\text{Successor}(x)))) \rightarrow \forall x\ F(x)$



Example 2: "ZFC axioms of set theory"

constant-names, function-names: none

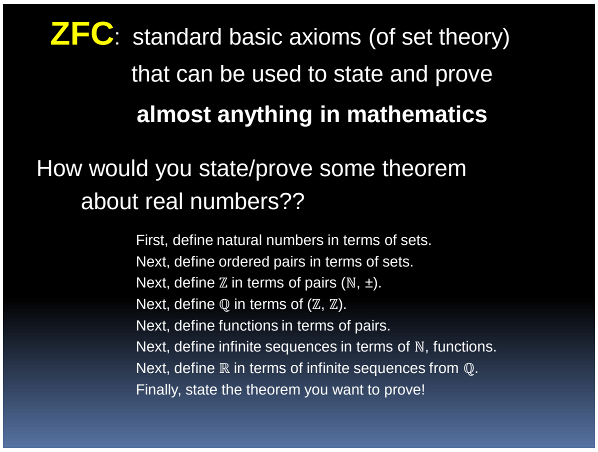
relation-name: IsElementOf(x,y)
["x∈y"]

extra axioms, catchily known as "ZFC":

$\forall x\ \forall y\ (\ (\forall z\quad z\in x \leftrightarrow z\in y)\ \rightarrow\ x = y\ )$

$\forall x\ \forall y\ \exists z\ (x\in z \wedge y\in z)$

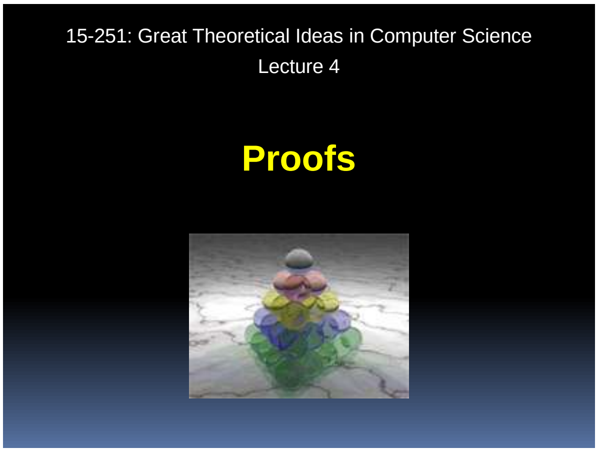
… 7 more (computable) axioms & schemas …



ZFC: standard basic axioms (of set theory) that can be used to state and prove **almost anything in mathematics**

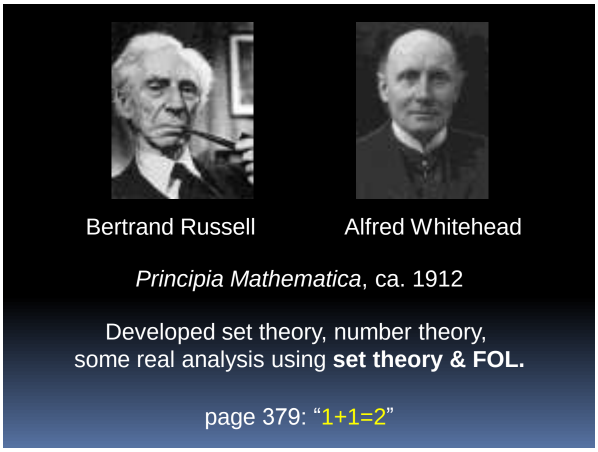
How would you state/prove some theorem about real numbers??

First, define natural numbers in terms of sets.
Next, define ordered pairs in terms of sets.
Next, define $\mathbb{Z}$ in terms of pairs ($\mathbb{N}$, ±).
Next, define $\mathbb{Q}$ in terms of ($\mathbb{Z}$, $\mathbb{Z}$).
Next, define functions in terms of pairs.
Next, define infinite sequences in terms of $\mathbb{N}$, functions.
Next, define $\mathbb{R}$ in terms of infinite sequences from $\mathbb{Q}$.
Finally, state the theorem you want to prove!



15-251: Great Theoretical Ideas in Computer Science
Lecture 4

Proofs



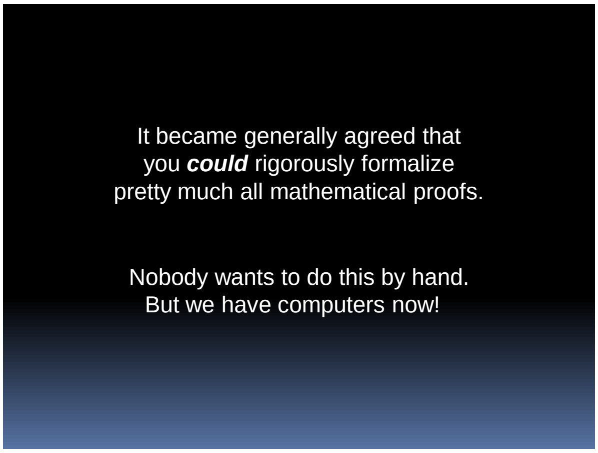
Bertrand Russell

Alfred Whitehead

Principia Mathematica, ca. 1912

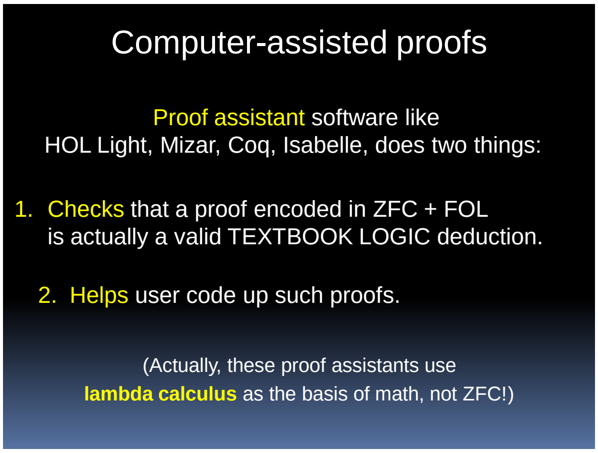
Developed set theory, number theory,
some real analysis using **set theory & FOL.**

page 379: “1+1=2”



It became generally agreed that
you ***could*** rigorously formalize
pretty much all mathematical proofs.

Nobody wants to do this by hand.
But we have computers now!

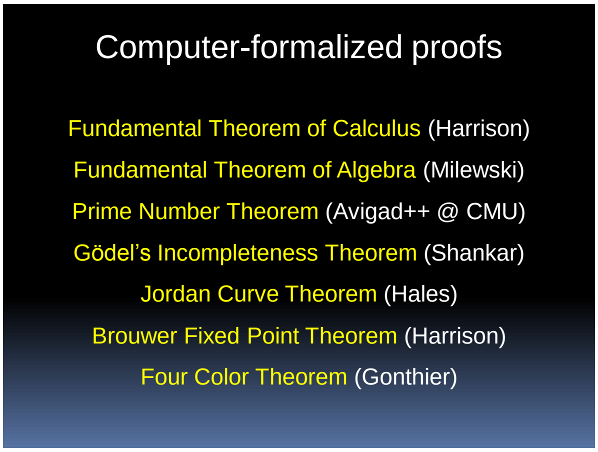


Computer-assisted proofs

Proof assistant software like
HOL Light, Mizar, Coq, Isabelle, does two things:

1. Checks that a proof encoded in ZFC + FOL
is actually a valid TEXTBOOK LOGIC deduction.
2. Helps user code up such proofs.

(Actually, these proof assistants use
lambda calculus as the basis of math, not ZFC!)



Computer-formalized proofs

Fundamental Theorem of Calculus (Harrison)

Fundamental Theorem of Algebra (Milewski)

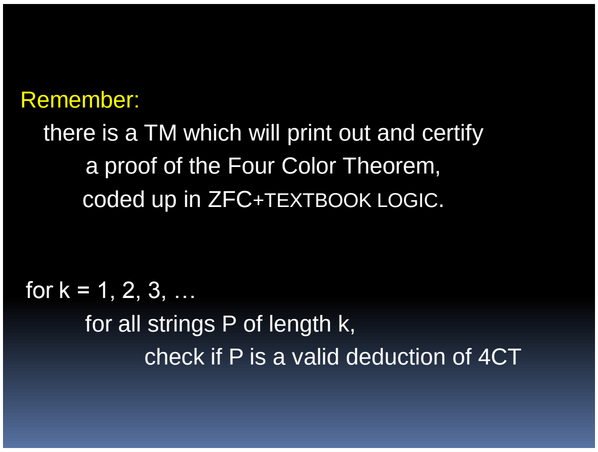
Prime Number Theorem (Avigad++ @ CMU)

Gödel’s Incompleteness Theorem (Shankar)

Jordan Curve Theorem (Hales)

Brouwer Fixed Point Theorem (Harrison)

Four Color Theorem (Gonthier)



Remember:
there is a TM which will print out and certify
a proof of the Four Color Theorem,
coded up in ZFC+TEXTBOOK LOGIC.

```
for k = 1, 2, 3, …
    for all strings P of length k,
        check if P is a valid deduction of 4CT
```

15-251: Great Theoretical Ideas in Computer Science
Lecture 21

Turing & Computability

Decidable languages

Definition:

A language $L \subseteq \Sigma^*$ is **decidable** if there is a Turing Machine M which:

1. **Halts on every input** $x \in \Sigma^*$.
2. Accepts inputs $x \in L$ and rejects inputs $x \notin L$.

The Halting Problem is Undecidable

Theorem:

Let $\text{HALTS} \subseteq \{0,1\}^*$ be the language
$\{ \langle M,x \rangle : \text{M is a TM which halts on input x} \}$.
Then HALTS is undecidable.

It's not: "we don't know how to solve it efficiently".

It's not: "we don't know if it's a solvable problem".

We know that it is unsolvable by any algorithm.

Proof

Assume M_{HALTS} is a decider TM which decides HALTS.

Here is the description of another TM called D, which uses M_{HALTS} as a subroutine:

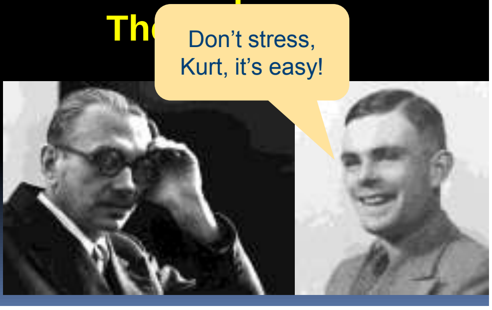
D:

Given as input $\langle M \rangle$, the encoding of a TM M:
D executes $M_{HALTS}(\langle M, \langle M \rangle \rangle)$.
If this call accepts, D enters an infinite loop.
If this call rejects, D halts (say, it accepts).

By definition, **$D(\langle D \rangle)$** loops if it halts and halts if it loops.
Contradiction.

15-251: Great Theoretical Ideas in Computer Science
Lecture 22

Gödel's Incompleteness Th

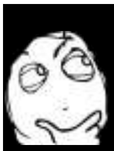


Suppose you just really cannot believe we proved that HALTS is undecidable.

How would you try to write a program H which, on input $\langle M,x \rangle$, decides if M(x) eventually halts?

Sample input:

$x = \epsilon$
M = "for k = 4, 6, 8, 10, 12, 14, …
if k is not the sum of 2 primes then HALT."



Dunno. Best idea I can think of is: Let H simulate M(x). If M(x) halts after 1,000,000,000 steps, output “it halts”. If M(x) still hasn’t halted after 1,000,000,000 steps, um…

How would you try to write a program H which, on input ⟨M,x⟩, decides if M(x) eventually halts?

Sample input:

$x = \epsilon$

```
M = “for k = 4, 6, 8, 10, 12, 14, …
        if k is not the sum of 2 primes then HALT.”
```

How would you try to write a program H which, on input ⟨M,x⟩, decides if M(x) eventually halts?

Idea for H:

```
“ for k = 1, 2, 3, …
    for all strings P of length k,
      • Check if P is a valid ZFC+TEXTBOOK LOGIC
        proof of the statement ‘M(x) eventually halts’
          If so, let H halt and output “yes, M(x) halts”
      • Check if P is a valid ZFC+TEXTBOOK LOGIC
        proof of the statement ‘M(x) eventually loops’
          If so, let H halt and output “no, M(x) loops”   ”
```

By my theorem: this TM H, like all algorithms, **does not** decide the Halting Problem

Idea for H:

```
“ for k = 1, 2, 3, …
    for all strings P of length k,
      • Check if P is a valid ZFC+TEXTBOOK LOGIC
        proof of the statement ‘M(x) eventually halts’
          If so, let H halt and output “yes, M(x) halts”
      • Check if P is a valid ZFC+TEXTBOOK LOGIC
        proof of the statement ‘M(x) eventually loops’
          If so, let H halt and output “no, M(x) loops”   ”
```

Conclusion:

There is some TM M and some string x such that ZFC+TEXTBOOK LOGIC ***cannot* prove** either of ‘M(x) eventually halts’ or ‘M(x) eventually loops’.

But M(x) either halts or it loops!
One of these two statements is true!

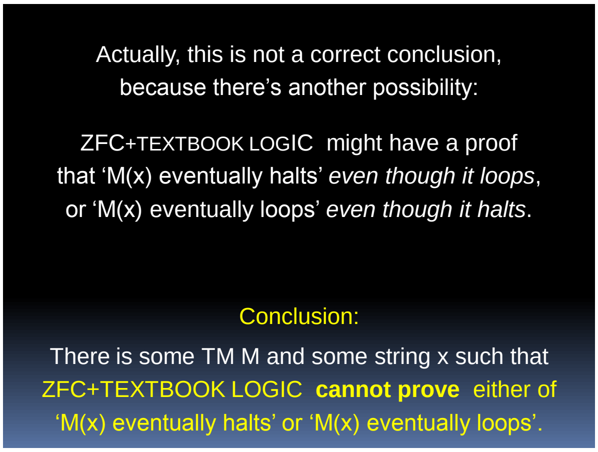
∴ There is a true mathematical statement that cannot be proved in ZFC.

This is basically
Gödel’s First Incompleteness Theorem.

```
“ for k = 1, 2, 3, …
    for all strings P of length k,
      • Check if P is a valid ZFC+TEXTBOOK LOGIC
        proof of the statement ‘M(x) eventually halts’
        If so, let H halt and output “yes, M(x) halts”
      • Check if P is a valid ZFC+TEXTBOOK LOGIC
        proof of the statement ‘M(x) eventually loops’
        If so, let H halt and output “no, M(x) halts”   ”
```

Conclusion:

There is some TM M and some string x such that ZFC+TEXTBOOK LOGIC ***cannot* prove** either of ‘M(x) eventually halts’ or ‘M(x) eventually loops’.

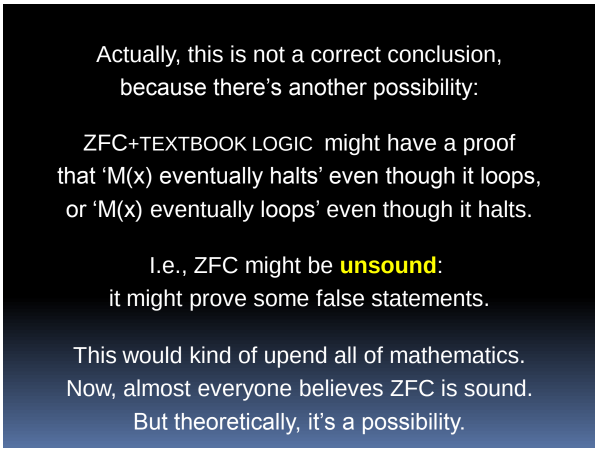


Actually, this is not a correct conclusion, because there's another possibility:

ZFC+TEXTBOOK LOGIC might have a proof that 'M(x) eventually halts' *even though it loops*, or 'M(x) eventually loops' *even though it halts*.

Conclusion:

There is some TM M and some string x such that ZFC+TEXTBOOK LOGIC **cannot prove** either of 'M(x) eventually halts' or 'M(x) eventually loops'.

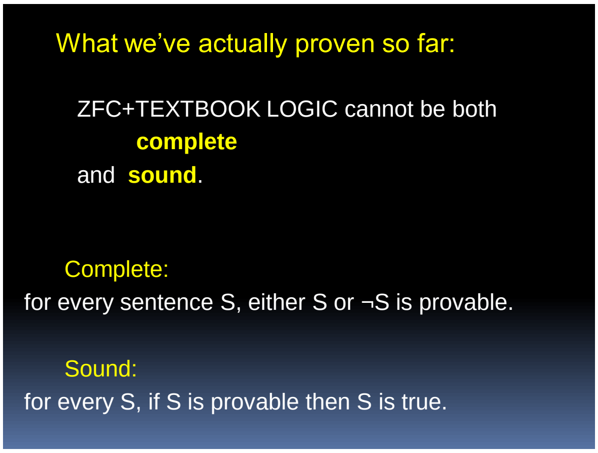


Actually, this is not a correct conclusion, because there's another possibility:

ZFC+TEXTBOOK LOGIC might have a proof that 'M(x) eventually halts' even though it loops, or 'M(x) eventually loops' even though it halts.

I.e., ZFC might be **unsound**: it might prove some false statements.

This would kind of upend all of mathematics. Now, almost everyone believes ZFC is sound. But theoretically, it's a possibility.

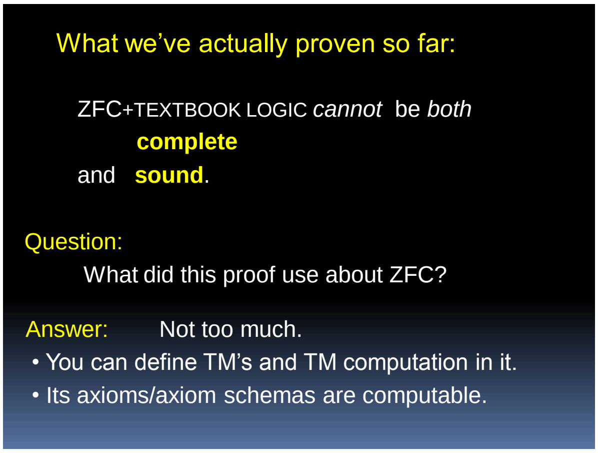


What we've actually proven so far:

ZFC+TEXTBOOK LOGIC cannot be both
complete
and **sound**.

Complete:
for every sentence S, either S or ¬S is provable.

Sound:
for every S, if S is provable then S is true.



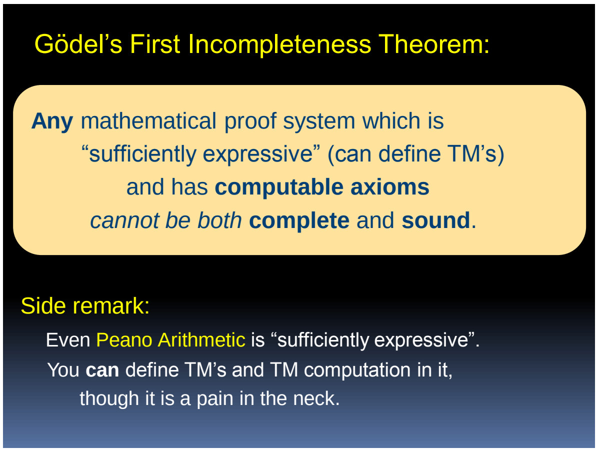
What we've actually proven so far:

ZFC+TEXTBOOK LOGIC *cannot* be *both*
complete
and **sound**.

Question:
What did this proof use about ZFC?

Answer: Not too much.

- You can define TM's and TM computation in it.
- Its axioms/axiom schemas are computable.

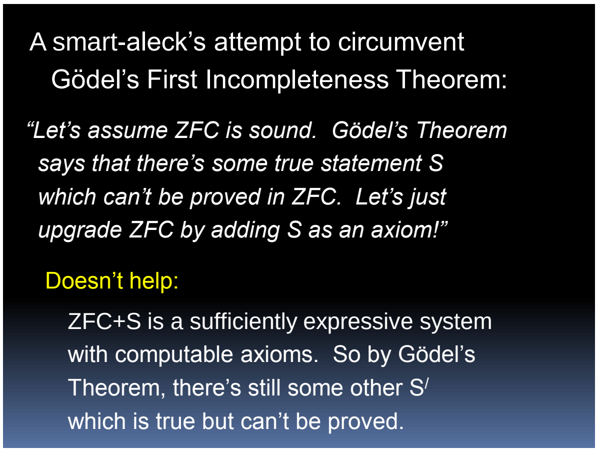


Gödel's First Incompleteness Theorem:

Any mathematical proof system which is "sufficiently expressive" (can define TM's) and has **computable axioms** *cannot be both* **complete** and **sound**.

Side remark:

Even Peano Arithmetic is "sufficiently expressive".
You **can** define TM's and TM computation in it, though it is a pain in the neck.



A smart-aleck's attempt to circumvent Gödel's First Incompleteness Theorem:

"Let's assume ZFC is sound. Gödel's Theorem says that there's some true statement S which can't be proved in ZFC. Let's just upgrade ZFC by adding S as an axiom!"

Doesn't help:

ZFC+S is a sufficiently expressive system with computable axioms. So by Gödel's Theorem, there's still some other S' which is true but can't be proved.

A smart-aleck's attempt to circumvent Gödel's First Incompleteness Theorem:

"Maybe add in $S^{/}$ as another axiom?"

Still doesn't help:

Apply Gödel's Theorem to ZFC+S+$S^{/}$,
get yet another true statement $S^{//}$ which
is true but cannot be proved.

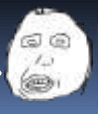
*"Maybe add in **all** true statements as axioms?"*

Okay fine, but now the set of axioms is not computable. So it's kind of a pointless system.

Gödel's First Incompleteness Theorem:

Any mathematical proof system which is
"sufficiently expressive" (can define TM's)
and has **computable axioms**
cannot be both **complete** and **sound**.
Equivalently, if it is sound, there are true statements
that are not provable within the system

How can you say a statement is true if you can't prove it?



Gödel Take 2

Just so that nobody gets confused, I'll prove an even stronger version which doesn't mention "truth".

Gödel's 1st: full version

(with strengthening by J. Barkley Rosser)

Any mathematical proof system which is
"sufficiently expressive" (can define TM's)
and has computable axioms
cannot be both **complete** and **consistent**.

Complete:
for every sentence S, either S or ¬S is provable.

Consistent:
for every S, you can't prove both S and ¬S.

Not only will we prove this,
there will be a bonus plot twist at the end!

For simplicity, we fix the mathematical proof system to be ZFC.

Outline of previous proof:

1. Assume ZFC sound.
2. Reason about a certain TM.
3. Deduce that ZFC is **incomplete**.

Outline of upcoming stronger proof:

1. Assume ZFC consistent.
2. Reason about a certain TM.
3. Deduce that ZFC is **incomplete**.

Lemma:

If a particular TM has a particular execution trace,
then there is a proof of this fact (in ZFC).

E.g., if M(x) halts, then there is a proof of ‘M(x) halts’.

Why? Can always write (in ZFC) proofs that look like:

“Initially M in the starting state/head/tape configuration.
After 1 step, M is in state/head/tape configuration *blah*.
After 2 steps, M is in state/head/tape configuration *blah*.
After 3 steps, M is in state/head/tape configuration *blah*.
. . . QED.”

Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

What can ZFC prove about D(⟨D⟩)? By consistency,
at most one of ‘D(⟨D⟩) halts’ or ‘D(⟨D⟩) loops’.

Perhaps ZFC can prove ‘D(⟨D⟩) loops’?

Then D on input ⟨D⟩ will find this proof, and thus halt.
But if D(⟨D⟩) halts **then ZFC can prove** ‘D(⟨D⟩) halts’ (by Lemma)
This contradicts consistency.

Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

By consistency, ZFC can prove
at most one of ‘D(⟨D⟩) halts’ or ‘D(⟨D⟩) loops’.

Perhaps ZFC can prove ‘D(⟨D⟩) halts’?

Then D(⟨D⟩) will run for some m steps, find this proof, and then execute the ‘infinite loop’ instruction. But then by the lemma, **there’s a proof of this fact** (the m+1 step execution trace). Thus ZFC can prove ‘D(⟨D⟩) loops’, contradicting consistency.

Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

Great! We just showed ZFC can prove neither
‘D(⟨D⟩) loops’ nor ‘D(⟨D⟩) halts’. So ZFC is incomplete. □

Incidentally… does D(⟨D⟩) **actually** halt or loop?

It loops. It does not find a proof of either statement.

Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

Great! We just showed ZFC can prove neither
‘D(⟨D⟩) loops’ nor ‘D(⟨D⟩) halts’. So ZFC is incomplete. □

Wait a minute.

It loops. It does not find a proof of either statement.

Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

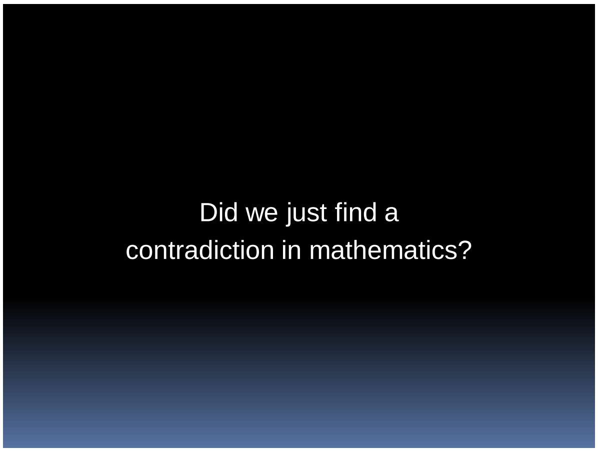
for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

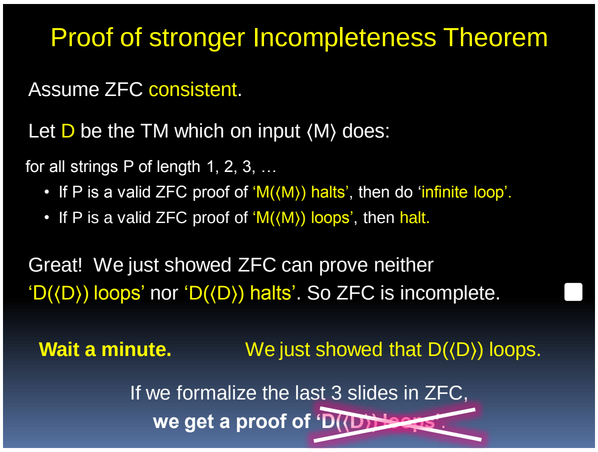
Great! We just showed ZFC can prove neither
‘D(⟨D⟩) loops’ nor ‘D(⟨D⟩) halts’. So ZFC is incomplete. □

Wait a minute. We just showed that D(⟨D⟩) loops!

If we formalize the last 3 slides in ZFC,
we get a proof of ‘D(⟨D⟩) loops’.



Did we just find a contradiction in mathematics?



Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

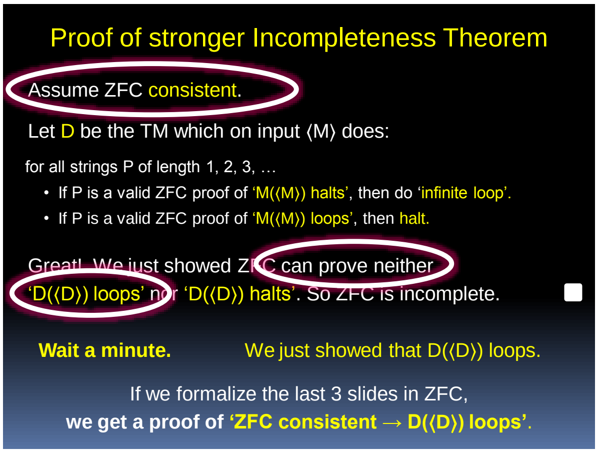
for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

Great! We just showed ZFC can prove neither ‘D(⟨D⟩) loops’ nor ‘D(⟨D⟩) halts’. So ZFC is incomplete. □

Wait a minute. We just showed that D(⟨D⟩) loops.

If we formalize the last 3 slides in ZFC,
we get a proof of ~~‘D(⟨D⟩) loops’~~.



Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

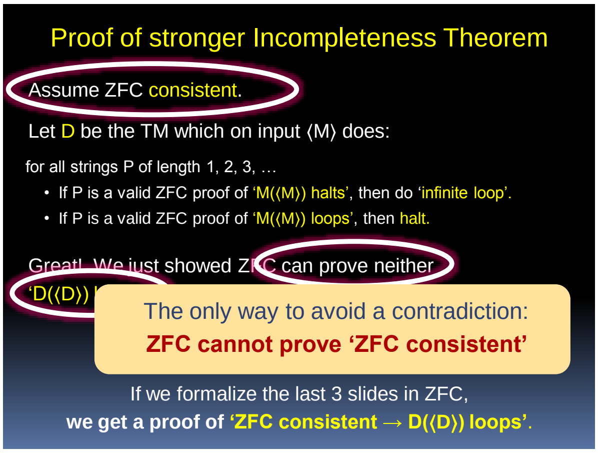
for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

Great! We just showed ZFC can prove neither ‘D(⟨D⟩) loops’ nor ‘D(⟨D⟩) halts’. So ZFC is incomplete. □

Wait a minute. We just showed that D(⟨D⟩) loops.

If we formalize the last 3 slides in ZFC,
we get a proof of ‘ZFC consistent → D(⟨D⟩) loops’.



Proof of stronger Incompleteness Theorem

Assume ZFC consistent.

Let D be the TM which on input ⟨M⟩ does:

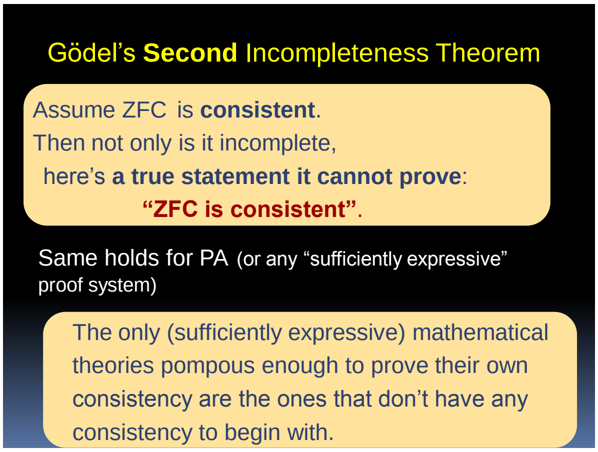
for all strings P of length 1, 2, 3, …

- If P is a valid ZFC proof of ‘M(⟨M⟩) halts’, then do ‘infinite loop’.
- If P is a valid ZFC proof of ‘M(⟨M⟩) loops’, then halt.

Great! We just showed ZFC can prove neither ‘D(⟨D⟩)

The only way to avoid a contradiction:
ZFC cannot prove ‘ZFC consistent’

If we formalize the last 3 slides in ZFC,
we get a proof of ‘ZFC consistent → D(⟨D⟩) loops’.

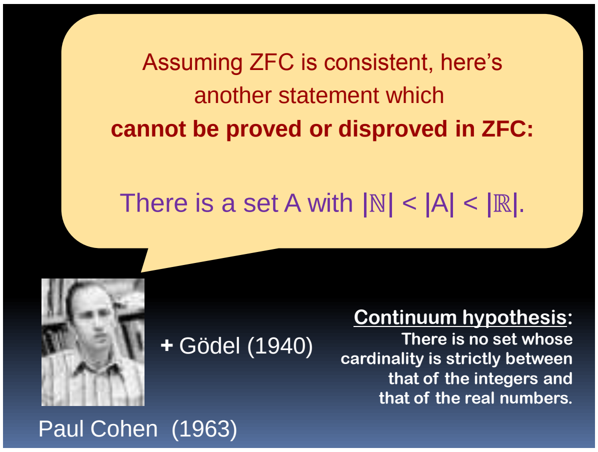


Gödel’s **Second** Incompleteness Theorem

Assume ZFC is **consistent**.
Then not only is it incomplete,
here’s **a true statement it cannot prove**:
“ZFC is consistent”.

Same holds for PA (or any “sufficiently expressive” proof system)

The only (sufficiently expressive) mathematical theories pompous enough to prove their own consistency are the ones that don’t have any consistency to begin with.



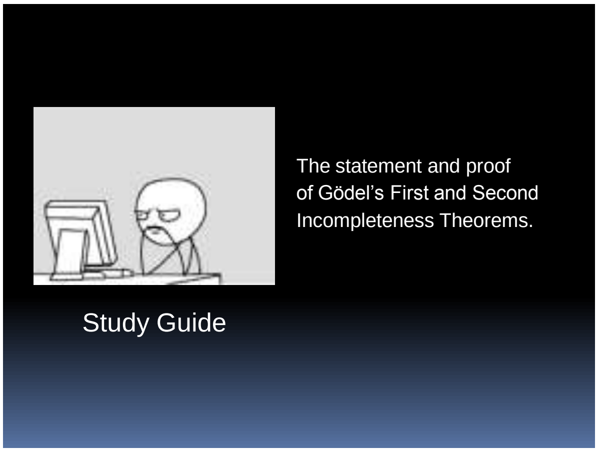
Assuming ZFC is consistent, here’s another statement which
cannot be proved or disproved in ZFC:

There is a set A with |ℕ| < |A| < |ℝ|.

+ Gödel (1940)

Paul Cohen (1963)

Continuum hypothesis:
There is no set whose cardinality is strictly between that of the integers and that of the real numbers.



The statement and proof of Gödel’s First and Second Incompleteness Theorems.

Study Guide