15-251: Great Theoretical Ideas in Computer Science
Lecture 16

Gödel's Incompleteness The

Proving the famous
"Gödel Incompleteness Theorems"
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
Lecture 3

Formalization of Proof

GORM: Good Old Regular Mathematics

- GORM is the math you've been doing all your life
- GORM is what we use in the lectures and homeworks
- GORM proofs are written in English (or another human language)
- In GORM, math statements are either true or false
  We try to prove the true ones, disprove the false ones
- GORM proofs are valid if they are:
  rigorous, logical, convincing, complete, precise
- This depends on the audience & assumed background!
- Ultimately, GORM proofs are valid if they are accepted by the community of mathematicians
- That's OK! But we may also want to try to formalize (within GORM) what it means to be a valid proof.

Formal proofs — 19th century

True rigor developed.

Culminated in the understanding that GORM proofs ***can*** be formalized, using tools like First Order Logic, & Deductive Systems.

First Order Logic:

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

Given a vocabulary, some sentences are "**tautologies**":

i.e., "true for all possible interpretations",
"automatically true, for 'purely logical' reasons".

e.g.:

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

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

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

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

"There's a (computable) FOL Deductive Calculus for tautologies."

This "FOL Deductive Calculus" has:

a bunch of axioms (initial objects),
(all of which are obviously tautologies);

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

Everything deducible is a tautology.
Gödel showed: every tautology is deducible.

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

"There's a (computable) FOL Deductive Calculus for tautologies."

Actually, FOL Deductive Calculus does not have finitely many axioms. It has finitely many "axiom schema". For example…

"if A is any sentence, then A∨¬A is an axiom"

"if IsR is any relation-name and c is any constant-name, then IsR(**c**)→(∃x IsR(x)) is an axiom"

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

"There's a (computable) FOL Deductive Calculus for tautologies."

"Computability":
There's an algorithm (say, a TM) which, given a sentence, decides if it is an axiom.

"if A is any sentence, then A∨¬A is an axiom"

"if IsR is any relation-name and c is any constant-name, then IsR(**c**)→(∃x IsR(x)) is an axiom"

Upshot of the Completeness Thm.

Corollary:

There is a TM algorithm which, given a **tautological** sentence S, finds a **deduction** of it in the FOL Deductive Calculus.

Proof:

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

(Cf. Midterm 1 Practice Problem #11: L = {S : S is a tautology} is 'tweetable'.)

Formalizing GORM proofs

1. Think of some universe you want to reason about.
2. Invent an appropriate vocabulary (constant, function, relation names).
3. Start with some axioms $A_1, \ldots, A_m$ which are true under the interpretation you have in mind.
4. See what theorems these axioms **entail**; i.e., for which T is $(A_1 \wedge \cdots \wedge A_m) \to T$ a tautology.

(By Gödel's theorem, equivalent to the T which you can **deduce** from the axioms using FOL Deductive Calculus.)

Ex. 1: Arithmetic of ℕ (Peano axioms)

constant-name: **0**

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

extra axioms:

∀x ¬(Successor(x)=**0**)
∀x ∀y (Successor(x)=Successor(y))→(x=y)
∀x Plus(x,**0**)=x
∀x ∀y Plus(x,Successor(y))=Successor(Plus(x,y))
∀x Times(x,**0**)=**0**
∀x ∀y Times(x,Successor(y))=Plus(Times(x,y),x)
"Induction:" For any parameterized formula F(x),
(F(**0**)∧(∀x F(x)→F(Successor(x)))) → ∀x F(x)

Getting ambitious: **All** of GORM??

In early 20th c., mathematicians sought a simple subject that could capture all GORM topics.

They came up with **Set Theory**.

It's extremely hacky and kludgy, but you **can** express all GORM concepts —
tuples, functions, naturals, integers, graphs, rationals, reals, calculus, *Turing machines*,
— with sets.

Ex. 2: Set Theory (ZFC axioms)

constant-names, function-names: none

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

extra axioms, catchily known as "ZFC":

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

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

… 7 more (computable) axioms & schemas …

Early 20th century conclusion:

You **can** formalize essentially all of GORM using ZFC + FOL Deductive Calculus.

However, it's super-painful to do by hand. (Russell & Whitehead page **379**: 1+1=2.)

But we have computers now…

Computer-assisted proof

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

1. Checks that a proof encoded in ZFC + FOL Deductive Calculus for First Order Logic (or typed lambda calculus theory) is valid.
2. Helps user code up such proofs.

Developing proof assistants is an active area of research, particularly at CMU!

Computer-formalized proofs

Fundamental Theorem of Calculus (*Harrison*)

Fundamental Theorem of Algebra *(Milewski)*

Prime Number Theorem *(Avigad @ CMU, et al.)*

Gödel's Incompleteness Theorem *(Shankar)*

Jordan Curve Theorem *(Hales)*

Brouwer Fixed Point Theorem *(Harrison)*

Four Color Theorem *(Gonthier)*

Feit-Thompson Theorem *(Gonthier)*

Kepler Conjecture *(Hales++)*

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

```
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 5

Turing's Legacy

15-251
Great Theoretical Ideas in Computer Science
Uncountability and Uncomputability

January 29th, 2015

Decidable languages

Definition:

A language $L \subseteq \Sigma^*$ is **decidable** (or computable) 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

Turing's Theorem:

Let HALTS $\subseteq \{0,1\}^*$ be the language
{ ⟨M,x⟩ : 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 ⟨M⟩, the encoding of a TM M:
D executes M_{HALTS}(⟨M, ⟨M⟩⟩).
If this call accepts, D enters an infinite loop.
If this call rejects, D halts (say, it accepts).

By definition, **D(⟨D⟩)** loops if it halts and halts if it loops.
Contradiction.

15-251: Great Theoretical Ideas in Computer Science
Lecture 16

Gödel's Incompleteness The

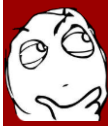
Suppose you just really cannot believe we proved that HALTS is undecidable.

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

Sample input:

M = "for k = 4, 6, 8, 10, 12, 14, …
check if k is the sum of 2 primes; if not, HALT"

x = ε (empty string)



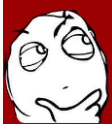
Dunno. Best idea I can think of is: Let H simulate M(x). If M(x) halts after 1,000,000,000 steps, output "it halts". If M(x) still hasn't halted after 1,000,000,000 steps, um…

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

Sample input:

M = "for k = 4, 6, 8, 10, 12, 14, …
check if k is the sum of 2 primes; if not, HALT"

x = ε (empty string)



I have a crazy and sort of awesome idea for how to write H.

Kurt, you mathematicians always make things too complicated.

Let me explain it.

All right, fine.

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+FOL Deductive Calculus 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+FOL Deductive Calculus 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+FOL Deductive Calculus 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+FOL Deductive Calculus 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+FOL Deductive Calculus **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+FOL Deductive Calculus).

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+FOL Deductive Calculus 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+FOL Deductive Calculus 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+FOL Deductive Calculus **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+FOL Deductive Calculus 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+FOL Deductive Calculus **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+FOL Deductive Calculus 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 + FOL Deductive Calculus 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 + FOL Deductive Calculus 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 severe 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**.

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

Whoahhhh, dude.
How can you say a statement S is true if you can't prove it?

Response 1

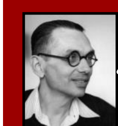
Don't get all confused. If I asked you yesterday,

"Hey, is it true that 1 is the only number which appears in Pascal's Triangle more than ten times?",

you wouldn't be, like,
"Whoahhhh dude, what does true mean?"

GORM doesn't suddenly become invalid just because you happen to be you're studying logic.

Response 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**.

We're going to need a lemma.

Some statements are so simple that, assuming they're true, they **definitely do** have a proof in ZFC.

Example: "There are 25 primes less than 100."

This definitely has a proof: the brute-force, brain-dead enumeration proof!

Our Brain-Dead Lemma:

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

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*.
... After t steps, M is in state/head/tape configuration *blah*.
QED."

In particular, if M(x) halts, there is a proof of 'M(x) halts'.

Outline of upcoming proof of the "truth"-free stronger version of Gödel's 1st:

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

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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a 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 Brain-Dead 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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a 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⟩) halts'?
Then D(⟨D⟩) will run for some t steps, find this proof, and then enter the 'go right forever' state. But by Brain-Dead Lemma, **there's a proof of this fact** (the t+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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a ZFC proof of 'M(⟨M⟩) loops', then halt.

Great! We just showed ZFC cannot prove either 'D(⟨D⟩) loops' or '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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a ZFC proof of 'M(⟨M⟩) loops', then halt.

Great! We just showed ZFC cannot prove either 'D(⟨D⟩) loops' or '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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a ZFC proof of 'M(⟨M⟩) loops', then halt.

Great! We just showed ZFC cannot prove either 'D(⟨D⟩) loops' or '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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a ZFC proof of 'M(⟨M⟩) loops', then halt.

Great! We just showed ZFC cannot prove either 'D(⟨D⟩) loops' or '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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a ZFC proof of 'M(⟨M⟩) loops', then halt.

Great! We just showed ZFC cannot prove either 'D(⟨D⟩) loops' or '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 ZFC proof of 'M(⟨M⟩) halts', enter 'go right forever' state.
- If P is a ZFC proof of 'M(⟨M⟩) loops', then halt.

Great! We just showed ZFC cannot prove either
'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'.

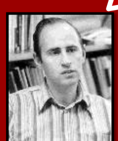
Gödel's **Second** Incompleteness Theorem

(proved independently by von Neumann)

Assume ZFC (or any "sufficiently expressive" proof system) is **consistent**. Then not only is it incomplete, here's **a true statement it cannot prove**: **"ZFC is consistent"**.

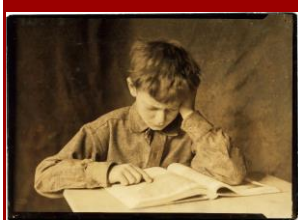
Assuming ZFC is consistent, here's another statement which **cannot be proved or disproved in ZFC:**

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



Paul Cohen (1963)

Study Guide



None.

You will not be tested on this topic.