CSCI3390-Lecture 9: Gödel's Theorem

October 2, 2018

In mathematics we prove things, and the things we prove are called theorems. Writing down exactly what constitutes a proof is a difficult task, and is the sort of thing that is gone to in great length in courses on Mathematical Logic. (In fact later in these notes we will go into it, but just a tiny bit.) But even without formal definitions, we can probably agree on some basic things we expect of proofs, at a minimum.

- *You should be able to check a proof by hand.* That is, a proof should lead you, step by little step, from its hypotheses to its conclusion, according to well-understood and agreed upon principles of deduction.
- *Statements of arithmetic that can be verified by a hand calculation are theorems.* By this we mean statements like $3^2 + 4^2 = 5^2$, and even the statement 'there exist positive integers x, y, z such that $x^2 + y^2 = z^2$'. For the latter, the verification that $3^2 + 4^2 = 5^2$ constitutes what we would ordinarily consider a proof of this fact.[1]
- *Theorems are true.* That is, you can't prove anything that's false.

In 1931, Kurt Gödel proved what is probably the most famous result of Mathematical Logic:

Theorem 1 *In any system satisfying the above criteria, there are true statements of arithmetic that are not theorems.*

[1] On the other hand, a *universally* quantified statement like 'for all positive integers x, y, z, $x^3 + y^3 \neq z^3$' would not have such a 'plug in and check' proof.

In other words, you can't prove everything that's true. There are some inherent limitations in mathematical systems—they simply cannot do everything we would expect them to do.

This has some profound implications for the philosophy of mathematics. Many people have seen in it implications for the philosophy of mind as well (often along the lines of 'people can do things that formal systems cannot'), but this is a matter of controversy.

1 The Forest: the proof in a nutshell

Gödel's Theorem is often viewed as one of those impossibly obscure things like quantum mechanics and general relativity, the understanding of which is past the reach of ordinary mortals. In computer science, it is often looked on as an *obvious* consequence of Turing's theory of undecidability, which was developed several years after Gödel's work. In fact, it follows as a direct corollary of:

Theorem 2 *The following problem is undecidable:*

> *Input: A statement ϕ of arithmetic.*
>
> *Output: Yes if ϕ is a theorem (i.e., can be proved), No otherwise.*

'Proof' of Theorem 2. Given a TM $\mathcal{M}$ and a string w, consider the statement '$\mathcal{M}$ accepts w'. If this is true, then we can verify it by hand, simulating $\mathcal{M}$ on w. This verification constitutes a proof, and thus '$\mathcal{M}$ accepts w' is a theorem. Conversely, if '$\mathcal{M}$ accepts w' is a theorem, then it must be true, because we cannot prove false things. We have thus reduced the problem L_{TM} to the decision problem in the theorem, so the latter is undecidable.

Proof of Theorem 1.Suppose, contrary to what we're trying to prove, that for every statement ϕ, either ϕ or its negation $\neg\phi$ is a theorem. Since we can check whether a given string is a valid proof, a TM can enumerate all strings over the alphabet in which we write our proofs, check for each of these strings whether it is a valid proof, and see if the last line of the proof is ϕ or $\neg\phi$. If it is ϕ the machine halts and accepts. If it is $\neg\phi$, the machine halts and rejects. By our assumption this machine halts and gives an answer for any ϕ, so we have an algorithm that decides whether ϕ is a theorem, contradicting Theorem 2. Thus there must be some statement ϕ such that neither ϕ nor $\neg\phi$ is a theorem.

Observe that the proof of Theorem 2 used the second and third of our informal requirements for proofs, and the the proof of Theorem 1 also used the first requirement. Observe too that the word *Proof* appears without quotation marks in the second argument. This was not an accidental omission—that really is the proof.

You can object that the argument proving Theorem 2 leaves something to be desired. We haven't really said what we mean by a statement of arithmetic, or a proof, in any concrete way. Furthermore, '$\mathcal{M}$ accepts w' doesn't really seem to be a statement about arithmetic, it's more a statement about a string-manipulation process. So we will look at a few more details below.

But, really, if you want to understand what Gödel's Theorem says and why it is true, the hand-wavy stuff we just did is more important than the somewhat more precise stuff that follows.

2 The Trees.

2.1 What is a statement of arithmetic?

Officially, it's a sentence of first-order predicate logic, which we interpret in the natural numbers $\mathbf{N}$. Such sentences use quantifiers, boolean connectives for and, or, and not, the constants 0 and 1, and symbols for addition, multiplication, powering, and order.[2] Just as an example

$$\forall x(\text{prime}(x) \rightarrow \exists y(\ \text{prime}(y) \wedge x < y))$$

where prime(z) means

$$(z > 1) \wedge \neg\exists x \exists y((x > 1) \wedge (y > 1) \wedge z = x \cdot y),$$

is a statement of arithmetic, which asserts that for every prime number there is a larger one–i.e., there are infinitely many primes.

2.2 What is a theorem? What is a proof?

Theorems are derived from other theorems by applying various rules of deduction, but we can't conjure up the *first* theorem out of nothing, so we have to begin with

[2] You really don't need to specify that order is built in to the language, since you can write $x \leq y$ as $\exists z(x + z = y)$. It is even possible to define $x^y = z$ in terms of the other relations, although this is considerably harder to do.

some starter theorems, which are called *axioms.* I will not provide you with a full list of axioms–the list varies from book to book—but here is a sampler.

$$\forall x(0 \leq x),$$

i.e., 0 is less than or equal to every number (remember we are working in the nonnegative integers) is an axiom.

For any statements ϕ and ψ,

$$(\phi \wedge \psi) \rightarrow \phi$$

is an axiom. This says: 'if ϕ and ψ are both true,then ϕ is true'. Most axioms are boring like these two.

For any formula $\phi(x)$ with a free variable x,

$$(\phi(0) \wedge \forall x(\phi(x) \rightarrow \phi(x+1))) \rightarrow \forall x \phi(x)$$

is an axiom. This is the principle of mathematical induction.

Observe that the last two examples each give infinitely many axioms (they are called *axiom schemas*) but we can tell whether a given formula has the required pattern, and the complete list of axioms and schemas is finite. Thus we can decide whether a given statement is an axiom—that is *the set of axioms is decidable*.

'Rules of deduction' are usually called *rules of inference*. There are a few of these, the exact number depending on the system used. A typical example is *modus ponens* which says 'from ϕ and $\phi \rightarrow \psi$, you can deduce ψ.'

A *proof* then, is a sequence of statements, where each statement is either an axiom, or can be deduced from the preceding statements by a rule of inference. As long as we can check whether a statement is an axiom and check that a rule of inference has been correctly applied, we can check whether a proof is correct. This was our first requirement for proofs, which we can restate: *The set of proofs is a decidable language.*

A *theorem* is the last statement in a proof. We are going to show that the set of theorems is *not* a decidable language, but we have an algorithm that semi-decides whether a statement is a theorem: To check if ϕ is a theorem, systematically generate all strings over the alphabet in which we write our formulas, check to see if this string is a proof, and answer Yes if the last statement in the string is ϕ. Thus *the set of theorems is a Turing-recognizable language.*

2.3 Gödelization: turning statements about Turing machines into statements about arithmetic.

In Gödel's original paper, he described a scheme for assigning numbers to the statements of arithmetic, and thus translating assertions about logic into statements of arithmetic. Since then, this process has been called *Gödel numbering* or *Gödelization*. Here we use it to assign numbers to Turing machines and their configurations.

We will suppose that the tape alphabet of all of our machines is $\{0, 1\}$, with 0 used as the blank symbol. We can simulate any Turing machine by a machine that satisfies this requirement, by encoding each tape symbol by several bits. (We really don't *need* this requirement, it just makes life simpler when we describe how to encode tape configurations.)

We represent states of a Turing machine by integers $0, 1, 2, \ldots$, with 0 reserved for the initial state and 1 for the accepting state. We can represent tape directions L and R by 0 and 1 respectively. We will then encode a transition, which is a quintuple

$$\tau = (q, \gamma, q', \beta, D)$$

by

$$<< \tau >> = 2^q \cdot 3^\gamma \cdot 5^{q'} \cdot 7^\beta \cdot 11^D.$$

For example, the integer 308 is the encoding of the transition rule 'in state 2, reading 0, write 1, enter state 0, and move right', because

$$308 = 2^2 \times 7 \times 11 = 2^2 \times 3^0 \times 5^0 \times 7^1 \times 11^1.$$

The specification of a Turing machine $\mathcal{M}$ is just a set $\{\tau_1, \ldots, \tau_k\}$ of such transition rules, which we encode by the integer

$$<< \mathcal{M} >> = 2^{<<\tau_1>>} 3^{<<\tau_2>>} \cdots p_k^{<<\tau_k>>},$$

where p_k is the k^{th} prime number. (Since the order of the transitions is irrelevant we actually get many different integers encoding the same Turing machine, but this does not pose a problem.) We call $<< \mathcal{M} >>$ the *Gödel number* of the Turing machine.

Thus a statement like, 'the Turing machine $\mathcal{M}$ in state 2 reading 0, writes 1, enters state 0, and moves right' translates into 'there is a prime p such that $p^{308} | m$ and $p^{309} \not| m$,' where m is the Gödel number of $\mathcal{M}$. Note that these encodings are huge numbers, even for very small Turing machines!

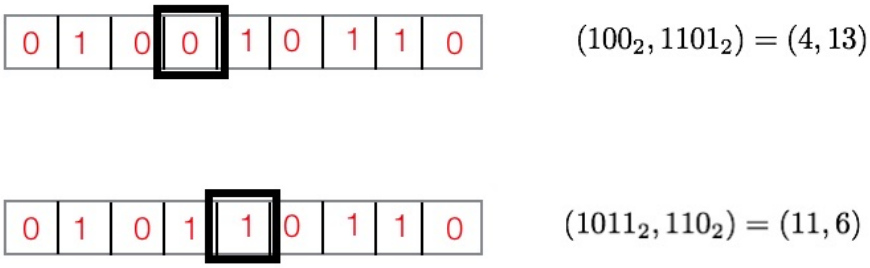

Figure 1: *Encoding the tape contents and current position of a Turing machine by a pair of integers. The operation performed by a Turing machine in a single step corresponds to some simple arithmetic with the pair of integers. In this case, the Turing machine has written 1 and moved one cell to the right.*

So we know how to encode states, transitions, and entire machines as integers. How do we encode tape contents and head positions? Figure 1 shows the scheme we use. The sequence of cells to the left of the head, including the cell currently scanned, is treated as the binary representation of a nonnegative integer x. Everything to the right of the head is the binary encoding (reading the bits with the most significant bit at the right, opposite to the usual direction) of another nonnegative integer y. Thus a configuration of the machine is encoded by a triple of integers (q, x, y), giving the state and these two values.

Manipulations of the tape contents and head are represented by very simple arithmetic operations: If, for example, we want to say, 'write 1 in the current position and move right', we are performing the operation

$$(x, y) \mapsto (4 * \left\lfloor \frac{x}{2} \right\rfloor + 2 + y \bmod 2, \left\lfloor \frac{y}{2} \right\rfloor).$$

(You can check that this gives the correct result for the example in the figure.)

2.4 Wrapping up

With all this in place, we can construct a formula that says 'the Turing machine with Gödel number x in configuration (q, y, z) transitions to configuration (q', y', z')'. Let's denote this formula, which has 7 free variables as

$$\phi(x, q, y, z, q', y', z').$$

It should be stressed that we have a *method for writing down this formula*. If we substitute specific values for the variables, we can verify by a hand calculation if it is true or not, and thus if it is true, by the criteria we introduced at the start, it can be proved. Proofs of Gödel's Theorem that go through all the details actually demonstrate this for the specific logical systems under consideration.

It is also possible to construct a formula with an additional free variable t that says, 'The Turing machine with Gödel number x in configuration (q, y, z) transitions to configuration $(q', y'.z')$ after exactly t steps.' We write this formula as

$$\psi(t, x, q, y, z, q', y', z').$$

The exact method for writing *this* formula is actually a bit tricky.

So, given a Turing machine $\mathcal{M}$ and a string w, we can compute the Gödel number m of $\mathcal{M}$, and the encoding (a, b) of the tape with tape contents w and the reading head positioned at the leftmost symbol of w. We can then write down the sentence

$$\exists t \exists y' \exists z' \psi(t, m, 0, a, b, 1, y', z').$$

This says, 'after some number t of steps, the machine $\mathcal{M}$ started in state 0 with w on its tape winds up in the accepting state 1,' or, more simply '$\mathcal{M}$ accepts w.' If $\mathcal{M}$ really does accept w, then the truth of this sentence can be verified with a direct calculation, so the sentence is a theorem, by the requirement we listed earlier. If $\mathcal{M}$ does not accept w, then the sentence cannot be a theorem, because what it asserts is false. Thus $\mathcal{M}$ accepts w if and only if the sentence we constructed is a theorem. This completes the reduction and the proof of Theorem 2. As we noted earlier, the proof of Theorem 1 does not change.

3 Perspective

The point of the hand-waving introduction to this lecture is that there is no way out of the difficulty that Gödel's theorem presents. It is not a property of a particular axiomatic system, but of *all* axiomatic systems that are powerful enough to allow you to describe and prove properties of Turing machines. You might be tempted to 'solve' the problem of a sentence ϕ that can be neither proved nor disproved by adding either ϕ or $\neg\phi$ as a new axiom. But that just kicks the can down the road–Gödel's theorem applies to this enlarged system as well, which consequently contains sentences that can be neither proved nor disproved.