A different formulation of the theorem

In the preceding section we proved that it is impossible to build a Turing Machine that outputs all and only the true statements of a rich enough arithmetical language.

That is one formulation of Gödel's Theorem.

Here is another:

> **Gödel's Incompleteness Theorem**
> No axiomatization of elementary arithmetic (in a rich enough language) can be both consistent and complete.

In this subsection we'll see what this means, and how it can be proved on the basis of the version of the result we have been discussing so far (assuming a particular understanding of "complete").

Axiomatization

Let me start with the notion of an axiomatization.

We've talked a fair amount about a language in which to express arithmetical claims. But we haven't said much about how to *prove* the arithmetical claims that are expressible in that language.

An **axiomatization** of arithmetic is a system for proving things within an arithmetical language. It consists of two different components: (a) a set of axioms, and (b) a set of rules of inference:

- An **axiom** is a sentence of the language that one chooses to treat as "basic", and on the basis of which other sentences of the language are supposed to be proved. One might, for example, treat "every number has a successor" (in symbols: "$\forall x_1 \exists x_2 \, (x_2 = x_1 + 1)$") as an axiom.

- A **rule of inference** is a rule for inferring some sentences of the language from others—a rule which is such as to guarantee that if one uses true sentences as input, one will always get true sentences as output. An example of a rule of inference is *modus ponens*: the rule that tells us that one can always infer the sentence ψ from the sentences ϕ and "if ϕ, then ψ".

In choosing an axiomatization, one is free to pick any list of axioms and rules of inference one likes, as long as one's list is Turing-computable. (In other words, it must be possible to program a Turing Machine to determine what counts as an axiom and when one sentence follows from another by a rule of inference.)

Provability, completeness and consistency

Once one has an axiomatization for a given language, one can introduce a notion of provability for that language.

A sentence S is **provable** on the basis of a given axiomatization if there is a finite sequence of sentences of the language with the following two properties:

- Every member of the sequence is either an axiom, or something that results from previous members of the sequence by applying a rule of inference.
- The last member of the sequence is S.

We can now say what it is for an axiomatization to be complete and consistent.

For an axiomatization to be **complete** is for every true sentence of the language to be provable on the basis of that axiomatization.

For it to be **consistent** is for it never to be the case that both a sentence of the language and its negation are provable on the basis of that axiomatization.

Notice that an arithmetical falsehood can never be proved on the basis of a complete and consistent axiomatization. To see this, suppose for *reductio* that a complete and consistent axiomatization does entail some false sentence S. Since S is false, its negation must be true. And since the axiomatization is complete, the negation of s must be provable on the basis of that axiomatization. So both a sentence and its negation must be provable on the basis of that axiomatization. So the axiomatization fails to be consistent.

(Warning: Gödel's Theorem is sometimes formulated using a slightly different notion of completeness, according to which an axiomatization is complete if it proves every sentence or its negation. The result holds on either characterization of completeness, but the version I

use here has the advantage of being easily provable on the basis of our original formulation of the theorem.)

The proof

Let us now use the claim that no Turing Machine can output all and only arithmetical truths (of a rich enough language) to prove that no axiomatization of arithmetic (in that language) can be both consistent and complete.

We proceed by *reductio*. We will assume that we have a consistent and complete axiomatization of arithmetic in a suitable language, and use this assumption to build a Turing Machine that outputs all and only the true sentences of that language. Since we know that there can be no such Turing Machine, our assumption must be false.

The first thing to note is that it is possible to code each finite sequence of sentences of our arithmetical language as a natural number. To see this, note that one can code each symbol of the language as a natural number. Since each sentence is a finite sequence of symbols, and since we know how to code finite sequences of natural numbers as natural numbers, this means that we can code each sentence as a natural number. And once sentences have been coded as natural numbers, we can repeat the process to code each finite sequence of sentences as a natural number.

The next observation is that one can program a Turing Machine that outputs all and only the sentences of our language that can be proved on the basis of our axiomatization. The machine proceeds by going through the natural numbers in order. At each stage, the machine determines whether the relevant number codes a sequence of sentences. If it doesn't, the machine goes on to the next number. If it does, the machine proceeds to determine whether the sequence constitutes a proof. (We know that this is possible because we know that the axioms and inference rules of our system are Turing-computable.) If the sequence does constitute a proof, the machine outputs the last member of the sequence (which is the conclusion of the proof). The machine then goes on to consider the next natural number.

Notice, finally, that if our axiomatization is consistent and complete, then a Turing Machine that outputs all and only the provable sentences of our arithmetical language must output all and only the true sentences of our arithmetical language. But we have shown this to be impossible for languages like L, which are rich enough to express the Halting Function. We conclude that an axiomatization for such a language cannot be both consistent and complete.

Discussion

Hide Discussion

Topic: Week 10 / A different formulation of the theorem

Add a Post

Show all posts by recent activity

? Is it possible to prove that a certain axiomatization is either consistent or complete? 2

Godel's theorem says that axiomatization of rich enough L can't be both complete and consisten...