Lecture 38

Gödel's Incompleteness Theorem

In 1931 Kurt Gödel [50, 51] proved a momentous theorem with far-reaching philosophical consequences: he showed that *no* reasonable formal proof system for number theory can prove all true sentences. This result set the logic community on its ear and left Hilbert's formalist program in shambles. This result is widely regarded as one of the greatest intellectual achievements of twentieth-century mathematics.

With our understanding of reductions and r.e. sets, we are in a position to understand this theorem and give a complete proof. It is thus a fitting note on which to end the course.

The Language of Number Theory

The first-order language of number theory L is a formal language for expressing properties of the natural numbers

$$\mathbb{N} = \{0, 1, 2, \ldots\}.$$

The language is built from the following symbols:

- variables $x, y, z, \ldots$ ranging over $\mathbb{N}$;
- operator symbols $+$ (addition) and $\cdot$ (multiplication);

D. C. Kozen, *Automata and Computability*

- constant symbols 0 (additive identity) and 1 (multiplicative identity);
- relation symbol $=$ (other relation symbols $<$, $\leq$, $>$, and $\geq$ are definable);
- quantifiers $\forall$ (for all) and $\exists$ (there exists);
- propositional operators $\vee$ (or), $\wedge$ (and), $\neg$ (not), $\rightarrow$ (if-then), and $\leftrightarrow$ (if and only if); and
- parentheses.

Rather than give a formal definition of the well-formed formulas of this language (which we could easily do with a CFG), let's give some examples of formulas and their interpretations.

We can define other comparison relations besides $=$; for example,

$$x \leq y \stackrel{\text{def}}{=} \exists z \; x + z = y,$$
$$x < y \stackrel{\text{def}}{=} \exists z \; x + z = y \wedge \neg(z = 0).$$

Many useful number-theoretic concepts can be formalized in this language. For example:

- "q is the quotient and r the remainder obtained when dividing x by y using integer division":

$$\text{INTDIV}(x, y, q, r) \stackrel{\text{def}}{=} x = qy + r \;\wedge\; r < y$$

- "y divides x":

$$\text{DIV}(y, x) \stackrel{\text{def}}{=} \exists q \; \text{INTDIV}(x, y, q, 0)$$

- "x is even":

$$\text{EVEN}(x) \stackrel{\text{def}}{=} \text{DIV}(2, x)$$

  Here 2 is an abbreviation for 1+1.

- "x is odd":

$$\text{ODD}(x) \stackrel{\text{def}}{=} \neg\text{EVEN}(x)$$

- "x is prime":

$$\text{PRIME}(x) \stackrel{\text{def}}{=} x \geq 2 \;\wedge\; \forall y \; (\text{DIV}(y, x) \rightarrow (y = 1 \vee y = x))$$

- "x is a power of two":

$$\text{POWER}_2(x) \stackrel{\text{def}}{=} \forall y \; (\text{DIV}(y, x) \wedge \text{PRIME}(y)) \rightarrow y = 2$$

- "y is a power of two, say 2^k, and the kth bit of the binary representation of x is 1":

$$\text{BIT}(x,y) \stackrel{\text{def}}{=} \text{POWER}_2(y) \wedge \forall q\, \forall r\, (\text{INTDIV}(x,y,q,r) \rightarrow \text{ODD}(q))$$

Here is an explanation of the formula $\text{BIT}(x,y)$. Suppose x and y are numbers satisfying $\text{BIT}(x,y)$. Since y is a power of two, its binary representation consists of a 1 followed by a string of zeros. The formula $\text{BIT}(x,y)$ is true precisely when x's bit in the same position as the 1 in y is 1. We get hold of this bit in x by dividing x by y using integer division; the quotient q and remainder r are the binary numbers illustrated. The bit we are interested in is 1 iff q is odd.

$$\begin{array}{rcl} y & = & 1000000000000 \\ x & = & \underbrace{11011001}_{q}\underbrace{0100011011}_{r} \end{array}$$

This formula is useful for treating numbers as bit strings and indexing into them with other numbers to extract bits. We will use this power below to write formulas that talk about valid computation histories of Turing machines.

If there are no free (unquantified) variables, then the formula is called a *sentence*. Every sentence has a well-defined truth value under its natural interpretation in $\mathbb{N}$. Examples are

$\forall x\, \exists y\, y = x + 1$ "Every number has a successor."

$\forall x\, \exists y\, x = y + 1$ "Every number has a predecessor."

Of these two sentences, the first is true and the second is false (0 has no predecessor in $\mathbb{N}$).

The set of true sentences in this language is called *(first-order) number theory* and is denoted $\text{Th}(\mathbb{N})$. The *decision problem* for number theory is to decide whether a given sentence is true; that is, whether a given sentence is in $\text{Th}(\mathbb{N})$.

Peano Arithmetic

The most popular proof system for number theory is called *Peano arithmetic* (PA). This system consists of some basic assumptions called *axioms*, which are asserted to be true, and some *rules of inference*, which can be applied in a mechanical way to derive further theorems from the axioms.

Among the axioms of PA, there are axioms that apply to first-order logic in general and are not particular to number theory, such as axioms for manipulating

- propositional formulas, such as $(\varphi \wedge \psi) \to \varphi$;
- quantifiers, such as $(\forall x\ \varphi(x)) \to \varphi(17)$; and
- equality, such as $\forall x\ \forall y\ \forall z\ (x = y \wedge y = z \to x = z)$.

In addition, PA has the following axioms particular to number theory:

$\forall x\ \neg(0 = x + 1)$	0 is not a successor
$\forall x\ \forall y\ (x + 1 = y + 1 \to x = y)$	successor is one-to-one
$\forall x\ x + 0 = x$	0 is an identity for $+$
$\forall x\ \forall y\ x + (y + 1) = (x + y) + 1$	$+$ is associative
$\forall x\ x \cdot 0 = 0$	0 is an annihilator for $\cdot$
$\forall x\ \forall y\ x \cdot (y + 1) = (x \cdot y) + x$	$\cdot$ distributes over $+$
$(\varphi(0) \wedge \forall x\ (\varphi(x) \to \varphi(x + 1))) \to \forall x\ \varphi(x)$	induction axiom

where $\varphi(x)$ denotes any formula with one free variable x. The last axiom is called the *induction axiom*. It is actually an axiom *scheme* because it represents infinitely many axioms, one for each $\varphi(x)$. It is really the induction principle on $\mathbb{N}$ as you know it: in words,

- if φ is true of 0 (basis), and
- if for any x, from the assumption that φ is true of x, it follows that φ is true of $x + 1$ (induction step),

then we can conclude that φ is true of all x.

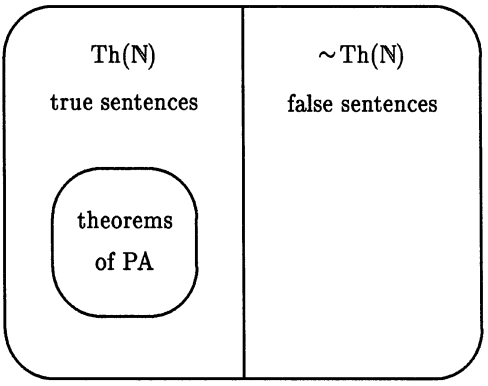
There are also two *rules of inference* for deriving new theorems from old:

$$\frac{\varphi \qquad \varphi \to \psi}{\psi}, \qquad \frac{\varphi}{\forall x\ \varphi}.$$

These two rules are called *modus ponens* and *generalization*, respectively.

A *proof* of φ_n is a sequence $\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$ of formulas such that each φ_i either is an axiom or follows from formulas occurring earlier in the list by a rule of inference. A sentence of the language is a *theorem* of the system if it has a proof.

A proof system is said to be *sound* if all theorems are true; that is, if it is not possible to prove a false sentence. This is a basic requirement of all reasonable proof systems; a proof system wouldn't be much good if its theorems were false. The system PA is sound, as one can show by induction on the length of proofs: all the axioms are true, and any conclusion derived by a rule of inference from true premises is true. Soundness means that the following set inclusions hold:



A proof system is said to be *complete* if all true statements are theorems of the system; that is, if the set of theorems coincides with Th(ℕ).