Goedel’s Theorem 7
A very little on Hilbert’s programme.

Goedel coding

The relation that holds between $m$ and $n$ when $m$ codes for a $PA$ derivation of the wff with code $n$ is p.r (and so is recursive).

The (standard) notation $\bar{\phi}$ to denote to code number for the wff $\phi$, and $\bar{\phi}^-$ to denote the standard numeral for $\bar{\phi}$.
Story so far

- The very idea of a formal axiomatised theory, the notion of negation incompleteness, and we stated (two versions of) Goedels First Theorem.
- Introduced ‘sufficient strength’ to prove an incompleteness theorem that is weaker than Goedels, since it doesn’t tell us how to construct a true-but-unprovable sentence.
- Specific axiomatisations of Arithmetic $BA$ and $Q$ which, although weak, had some interesting properties. $BA$ is complete, the stronger $Q$ is not.
Story so far

- The axiom of induction which strengthens $\mathbb{Q}$ to $PA$.
- To get more than just addition and multiplication we looked at the recursive functions.
- $\mathbb{Q}$ not only can express but also capture all the recursive functions.
Hilbert’s Programme

- Although Hilbert’s predecessors had a concept of axiomatisation, mostly they were ‘contentual’.
- Hilbert hoped to be able to put any branch of mathematics, e.g. analysis, on firm footing by providing a ‘formal’ axiomatisation of it.
- Once axiomatised, content becomes redundant and you are left with a symbolic game, provided it is consistent.
A formal theory can now be subject to a formal consistency proof.

Hilbert hoped that finitary means are enough to prove all required consistency results.

Roughly, a system is finitary if any explicit (e.g. set) or implicit (e.g. quantifier) reference to infinitary entities can be eliminated.
We describe a coding of the syntax that allows us express the relation: ‘$x$ codes a wff that expresses a property that $y$ derivably satisfies’.

First we code the individual symbols:

| ¬ | ∧ | ∨ | → | ∀ | ∃ | = | ( | ) | 0 | s | + | × | $x_1$ | $x_2$ | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 | 2 | 4 | ... |
Arithmetisation of syntax

- Given a symbol $\sigma$ let $\overline{\sigma}$ be its code.
- Let expression $e$ be the sequence of $k$ symbols and/or variables $\sigma_1, \sigma_2, \ldots, \sigma_k$. Also, for $i > 0$ let $p_i$ be the $i$-th prime number. Then $e$'s Godel number (g.n.) $\overline{e}$ is:

$$\prod_i p_i^{\overline{\sigma_i}}$$

i.e.

$$2^{\overline{\sigma_1}} \times 3^{\overline{\sigma_2}} \times 5^{\overline{\sigma_3}} \times \cdots \times p_k^{\overline{\sigma_k}}$$
Goedel numbers

- If $\phi$ is an L-expression, then we use $\overline{\phi}$ to denote the Goedel number of $\phi$.
- We use $\overline{\overline{\phi}}$ to denote the numeral of the Goedel number of $\phi$.
- Notice that $\overline{\overline{\neg \neg \neg}} = 2^1 \times 3^1 = 6 = \overline{\chi_3 \neg}$.
- Notice that $\overline{\phi}$ is always larger than the length of $\phi$ or the goedel number of any symbol in $\phi$. 
Coding sequences

- Given a sequence $\alpha$ of wffs or other expressions $\xi_1, \ldots, \xi_n$, we obtain a single ‘super’ g.n. $\top \alpha \bot$ for the sequence:

$$\prod_i p_i^{\top \xi_i \bot}$$

- So when we decode a super g.n., we first decode it into a sequence of g.n.s and then decode those.
Having fixed on our scheme of Goedel numbering, define the following numerical properties:

1. $\text{Term}(n)$ is to hold when $n$ codes for a term of $L_A$.
2. $\text{Wff}(n)$ is to hold when $n$ codes for a wff of $L_A$.
3. $\text{Sent}(n)$ is to hold when $n$ codes for a closed sentence of $L_A$.

We must show $\text{Trm}(n)$, $\text{Wff}(n)$, and $\text{Sent}(n)$ are recursively decidable properties (in fact, they are p.r.).
Logical operations

- If $c_P$ and $c_F$ are characteristic functions for $P$ and $F$, then we can form characteristic functions for $\neg P$ and $P \land F$:

$$z(c_P(x)) \quad c_P(x) \times c_F(x)$$

This allows us to construct characteristic functions from any combination of properties/relations by propositional connectives and bounded quantification.
Logical operations

- Definitions by cases

\[
f(x) = \begin{cases} 
  g(x) & \text{if } Px \\
  h(x) & \text{otherwise}
\end{cases}
\]

are handled by:

\[
f(x) = (c_p(x) \times g(x)) + (\neg c_p(x) \times h(x))
\]

- We can now show many properties are recursive, in fact p.r., by encoding their logical definitions:

\[
Prm(n) \text{ iff } n \neq 1 \land \forall x \leq n \forall y \leq n (x \times y = n \rightarrow (x = 1 \lor y = 1))
\]
A coding scheme $S \ L_A$ mapping expressions to numbers is acceptable iff there is a recursive function which ‘translates’ code numbers according to $S$ into code numbers under our official Goedelian scheme, and another recursive function which converts code numbers under our scheme back into code numbers under scheme $S$.

A property like $Trm$ defined using our official Goedelian coding scheme is recursive if and only if the corresponding property $Trm_S$ defined using scheme $S$ is recursive, for any acceptable scheme $S$. 
We have to show that this is recursive:

- $n$ is the last in a sequence of sequences symbols each of which is 0 or a variable or is the concatenation of $s$, + or $\times$ (and brackets) with one/two earlier members of the sequence.

‘a sequence of symbols’ is actually a bounded quantification as the Goedel number of the sequence cannot be any greater than $(p^n_l)^l$ where $l$ is the length of $n$.

- In fact we can show that it is p.r.
Showing this is is not hard, but tedious. Similarly we can show that $Wff(n)$, and $Sent(n)$ ar p.r.

We can also then show that, if $Z$ is an axiomatised theory, then $Prf_Z(n, m)$ is recursive and so can be expressed by $L_A$ (and captured by $Q$).

If $PA \vdash \forall x \neg Prf_Z[x, \square \bot]$ then $PA$ has derived a sentence expressing that $Z$ is consistent.

Probably all axiomatised theories we will ever work are p.r. axiomatisable.