• **Computability** graduated to a **theory** only in the 20th Century —

• The direction of this being initially determined by the famous mathematician **David Hilbert** (born Königsberg 1862 – died Göttingen 1943).

•• In 1900 — Hilbert set an agenda for 20th Century mathematics, at the International Congress of Mathematicians in Paris —

• Listing 23 “**particular definite problems, drawn from various branches of mathematics, from the discussion of which an advancement of science may be expected**”. In particular —

•• **Hilbert’s Tenth Problem** — “**to devise a process according to which it can be determined by a finite number of operations whether [a Diophantine] equation is solvable in rational integers**” —
Where a *diophantine equation* is a polynomial with integer coefficients, e.g.:

\[
x^4 - 3x^3 + 5x^2 - 7x - 6 = 0 \quad (1)
\]
\[
x^2 + y^2 = z^2 \quad (x, y, z > 0) \quad (2)
\]
\[
x^3 + y^3 = z^3 \quad (x, y, z > 0) \quad (3)
\]
\[
x^2 - 2y^2 = 0 \quad (x, y > 0) \quad (4)
\]
\[
x^3 y + 4y^2 z^2 - 7xyz + 12x - 11y + 14 = 0 \quad (5)
\]

**Note:** No *obvious* general algorithm exists for telling whether the above equations have integer solns. —

• (1), (3) and (4) do not have solns. — but for different reasons —

• (1) uses the Rational Root Test — (3) can be seen to be a special case of Fermat’s Last Theorem (due to Euler, 1770) — And (4) depends on \( \sqrt{2} \) being irrational.

• While (2) has many solns. — e.g., \((3, 4, 5)\) — and what about (5)??
**Question:** If there seems no valid approach to getting a positive solution to Hilbert’s Tenth Problem — How would one go about getting a negative solution?

**More generally:** Are there unsolvable classes of problems in mathematics? And what limits are there on the ultimate capabilities of computers? And what relevance has this to human minds?

•• 1930s — The notion of *computability* and of *algorithm* theoretically captured in work by **Kurt Gödel**, **Alonzo Church**, **Stephen Kleene** — and **Alan Turing** (1912–1954).

•• 1936 — Invention of the *Universal Turing machine* — and the discovery of basic unsolvable problems.

•• 1970 — Negative solution to Hilbert’s Tenth Problem (Davis, Matyasevich, Putnam and Julia Robinson).
• A URM has *registers* which store non-negative integers:

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
r_1 & r_2 & r_3 & r_4 & \cdots \\
\hline
R_1 & R_2 & R_3 & R_4 \\
\hline
\end{array}
\]

• A URM *program* is a finite sequence of instructions, each of which is one of 4 basic types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Symbolism</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>(Z(n))</td>
<td>(r_n = 0)</td>
</tr>
<tr>
<td>successor</td>
<td>(S(n))</td>
<td>(r_n = r_n + 1)</td>
</tr>
<tr>
<td>transfer</td>
<td>(T(m,n))</td>
<td>(r_n = r_m)</td>
</tr>
<tr>
<td>jump</td>
<td>(J(m,n,q))</td>
<td>If (r_n = r_m) go to instruction (q) — else go to next instruction</td>
</tr>
</tbody>
</table>

• **Input convention:** Input \((x_1,\ldots,x_n)\) by starting with \(x_1,\ldots,x_n\) in registers \(R_1,\ldots,R_n\), resp'y, and 0 in the other registers.

• **Output convention:** If a computation halts, the output is the number in register \(R_1\) — there is no output otherwise.
• The URM program $P$ computes the function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ iff — for all $(x_1, \ldots, x_k) \in \mathbb{N}^k$ — the computation with input $(x_1, \ldots, x_k)$ using program $P$ halts with output $f(x_1, \ldots, x_k)$.

• $f$ is URM-computable iff there is a URM program which computes $f$.

\begin{center}
\begin{tabular}{l}
\textbf{Theorem 1.1 — The Basic URM-Computable Functions:} \\
The following are URM-computable: \\
(a) $z : n \mapsto 0$ (the zero function) \\
(b) $s : n \mapsto n + 1$ (the successor function) \\
(c) $U^k_i : (n_1, \ldots, n_k) \mapsto n_i$ for $1 \leq i \leq k$ (the projection functions)
\end{tabular}
\end{center}
§3 Closure properties

- $\ell(P) = $ the number of instructions in $P$.
- $\rho(P) = $ the largest index $k$ of a register $R_k$ used by $P$.
- $P$ is in *standard form* if in every $J(m, n, q)$ of $P$ with $q > \ell(P)$, have $q = \ell(P) + 1$.
- The *join* of programs $P, Q$ is the program

$$
P
\quad
Q$$

got by writing the instructions of $Q$, any $J(m, n, q)$ replaced by $J(m, n, \ell(P) + q)$, after those of $P$.

**Theorem 3.1:** The URM computable functions are closed under composition:

If $f : \mathbb{N} \to \mathbb{N}$, $g : \mathbb{N} \to \mathbb{N}$ are URM computable then so is $f \circ g : n \mapsto f(g(n))$.

**Theorem 3.2:** If $f : \mathbb{N} \to \mathbb{N}$, $g : \mathbb{N} \to \mathbb{N}$, $h : \mathbb{N}^2 \to \mathbb{N}$ are URM computable then so is the function $n \mapsto h(f(n), g(n))$. 
• Write $\vec{x}_k$ for $x_1, \ldots, x_k$.

• Let $h : \mathbb{N}^s \to \mathbb{N}$ be a function of $s$ variables, and for $1 \leq i \leq s$ let $f_i : \mathbb{N}^t \to \mathbb{N}$ be a function of $t$ variables. Then say $g : \mathbb{N}^t \to \mathbb{N}$ defined by

$$g(\vec{x}_t) = h(f_1(\vec{x}_t), \ldots, f_s(\vec{x}_t))$$

is obtained from $h, f_1, \ldots, f_s$ by substitution.

**Theorem 3.3:** The URM computable functions are closed under substitution.

•• Let $g : \mathbb{N}^k \to \mathbb{N}$, $h : \mathbb{N}^{k+2} \to \mathbb{N}$ be functions. Say that $f : \mathbb{N}^{k+1} \to \mathbb{N}$ defined by the **primitive recursive scheme**

$$f(\vec{x}_k, 0) = g(\vec{x}_k)$$

$$f(\vec{x}_k, y + 1) = h(\vec{x}_k, y, f(\vec{x}_k, y))$$

is obtained from $g, h$ by **primitive recursion**.

**Theorem 3.4:** The URM computable functions are closed under primitive recursion.
A function is *primitive recursive* if it can be obtained from the basic functions $z, s, U_i^k$ by a finite number of substitutions and primitive recursions.

**Theorem 3.5**: All the primitive recursive functions are URM-computable.

The function $f : \mathbb{N}^k \to \mathbb{N}$ is obtained from $g : \mathbb{N}^{k+1} \to \mathbb{N}$ by *minimalisation* or *μ-operator* if

$$f(x_k, 0) = \mu y[g(x_k, y) = 0]$$

= the least $y$ such that $g(x_k, y) = 0$ if there is one — undefined otherwise.

- $f : \mathbb{N}^k \to \mathbb{N}$ is *total* if its domain is the whole of $\mathbb{N}^k$ — and otherwise is *partial*.

- A total function is *recursive* if obtainable from $z, s, U_i^k$ by a finite number of substitutions, primitive recursions and minimalisations.

**Theorem 3.6**: All recursive functions are URM-computable.
§4 The Church-Turing Thesis

THEOREM 4.1: A function is URM computable function ⇔ it is recursive.

CHURCH-TURING THESIS

• A function is computable in the intuitive sense if — and only if — it is recursive/URM-computable.
§5 Non-Computable Functions

• If $f$, $g$ are functions, say that $g$ dominates $f$ if for some $n_0 \in \mathbb{N}$, $n > n_0 \Rightarrow g(n) > f(n)$.

• If $S$ is a set of functions, say that $g$ dominates $S$ if $g$ dominates every function in $S$.

• $f$ is strictly increasing if for all $n_1, n_2 \in \mathbb{N}$, $n_1 < n_2 \Rightarrow f(n_1) < f(n_2)$.

**Lemma 5.1:** Every URM computable function is dominated by a strictly increasing URM computable function.

• The **Busy Beaver function** is defined by:

$$B(n) = \text{the maximum output, for input 0, of any URM program with at most } n \text{ instructions.}$$
**Lemma 5.2:** $B$ is strictly increasing.

**Lemma 5.3:** For all $n \geq 1$, $B(n + 5) \geq 2n$.

**Theorem 5.4** (Tibor Radó, 1962):
$B$ dominates every URM computable function.

**Corollary 5.5:** The function $B$ is not URM computable.