

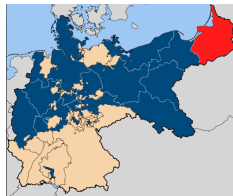
The Road from Leibniz to Turing, Part II

DA2205, DD3001

October 11, 2012

Hilbert to the Rescue

- **Born:** Königsberg, Prussia, Jan 1862.
- **Died:** Göttingen, Germany, Feb 1943.



- **Education:**

- ★ University of Königsberg, 1880–1885
- ★ Awarded PhD in Mathematics in 1885

- **Employment:**

- ★ Privatdozent, University of Königsberg, 1886–1892
- ★ Professor, University of Königsberg, 1892–1895
- ★ Chairman of Mathematics, University of Göttingen, Germany, 1895–1943



David Hilbert: Major Research Achievements

- Discovered and developed a broad range of fundamental ideas in many areas
 - ★ invariant theory
 - ★ the axiomatization of geometry
 - ★ theory of Hilbert spaces – one of the foundations of functional analysis.
- Defined direction of mathematical research with his list of 23 problem **Hilbert's problems** published in 1900.
- One of the founders of proof theory and mathematical logic.
- Among the first to distinguish between mathematics and metamathematics.



David Hilbert: Fame & fortune during lifetime

- Recognized as one of the leading mathematicians of his generation.
- Was a leader in the field and set the agenda for much of the research completed.
- Made “honorary citizen” of Königsberg in recognition of his achievements.
- Head of the Mathematics Dept. of University of Göttingen during its glory days.
- But, saw this Department decimated by the Nazis and World War II.

Early Triumphs: Proving Gordan's conjecture

- **1888:** Hilbert published an **existence** proof of Gordan's Conjecture – a major mathematical problem of the day.
- **Approach:** Supposing Gordan's Conjecture is false lead to a contradiction.
- Gordan's reaction
“ This is not mathematics. This is Theology”
(Quote highlights the divide in mathematics at the time.)
- His existence proof \longrightarrow Hilbert to a constructive proof.

Two Views of Existence in Mathematics

View 1:

- Kronecker - mathematical proofs of *existence* must be **constructive**.
- ...that is provide a method to construct the ideas in question.
- Brouwer maintained and advanced these thoughts:

“to exist in mathematics means to be constructed by intuition and the question whether a certain language is consistent, is not only unimportant in itself, it is also not a test for mathematical existence.”

- Due to these ideas opposed to
 - ★ general notion of irrational numbers
 - ★ using the logical law of the excluded middle for inference
 - ★ Cantor's transfinite numbers
 - ★ using the theorem that an infinite set of natural numbers always contains a least

View 2

- Hilbert believed existence proofs were fine.
- ...prove the existence of a mathematical entity required
a proof that assuming the existence of such an entity would not lead
to contradictions.
- One could rely on all the mathematical methods, tools and concepts that had developed over the years.

Early Triumphs: Axiomatization of Geometry

- **1898:** Hilbert published his *Axiomatization of Euclidean Geometry*.
- Hilbert's approach signaled his interest in the foundations of mathematics.
- Theorems deduced from the axioms using pure logic.
 - ★ Emphasized the abstract nature of the subject.
 - ★ Avoided the corrupting influence of diagrams!
- Provided proof his axioms of geometry consistent \iff arithmetic of real numbers consistent.

- **Venue:** International Congress of Mathematicians in Paris, August 1900
- Hilbert presents, as a challenge to his colleagues, 23 problems that seemed utterly inaccessible by the methods available at the time.
- These included
 - ★ Deciding the truth of Cantor's Continuum Hypothesis.
 - ★ Establishing the consistency of the axioms for the arithmetic of real numbers.
 - ★

Note this presentation was before the announcement of Russell's paradox in 1902.

- At the International Congress of Mathematicians of 1904
- Foundations of mathematics is in crisis
 - ★ Russell's paradox has thwarted Frege's aim to deduce mathematics from logic.
 - ★ Cantor's transfinite numbers make many uneasy
- Hilbert outlines the form a consistency proof for arithmetic may take.
- However, Hilbert is guilty of circular reasoning.

tools used for proof are the tools he wants to justify
- Hilbert abandons his pursuit of consistency until the 1920's.

Principia Mathematica, Russell & Whitehead 1910-1913

- Publication shows with simple direct steps can go:
Frege's pure logic \longrightarrow mathematics
- Avoids Russell's paradox, but in a cumbersome way.
- ✓ Complete formalization of mathematics in a symbolic logic is feasible.
- Issue of consistency of the entire structure **could not be** considered as system co-mingled language of logic and mathematics.
- Poincaré noted that if one took Russell's efforts seriously then it opened up the possibility of reducing mathematics to mere computation.

How to approach the foundations of mathematics?

- **Formalism**

- Hilbert & co.
- They wanted to ground mathematics on a small basis of a logical system proved sound by *metamathematical finitistic* means – construction from the natural numbers in a finite number of steps

- **Intuitionism**

- Brouwer & co.
- An approach to mathematics as the constructive mental activity of humans
- They discarded formalism as a meaningless game with symbols.

- In the 1920s Hilbert attacked the problem of the *consistency of arithmetic* with gusto.
- Did so with help of student Ackermann, assistant Bernays and John von Neumann.
- Hilbert's idea was to introduce a brand-new kind of mathematics called *proof theory* or *metamathematics*.
- The consistency proof was to be carried within *metamathematics*.
- This goal led to Hilbert's program...

Providing a secure foundations for mathematics for Hilbert required

- **Formalization:** all mathematical statements should be written in a precise formal language, and manipulated according to well defined rules and have finite set of axioms.
- **Completeness:** a proof that all true mathematical statements can be proved in the formalism.
- **Consistency:** a proof that no contradiction can be obtained in the formalism of mathematics. This consistency proof should preferably use only "*finitistic*" reasoning about finite mathematical objects.
- **Conservation:** a proof that any result about "*real objects*" obtained using reasoning about "*ideal objects*" (such as uncountable sets) can be proved without using ideal objects.
- **Decidability:** there should be an algorithm for deciding the truth or falsity of any mathematical statement.

Hilbert's Challenge & Catastrophe

- At the International Conference of Mathematicians in Bologna 1928 set the challenge
 - ★ Given a formal system where Frege's first-order logic is applied to a system of axioms for the natural numbers – Peano arithmetic (PA).
 - ★ Prove that PA is complete.
- Two years later **Kurt Gödel** solved this problem and his resolution was not what Hilbert expected.
- Announced his results at conference in Königsberg in honour of Hilbert's retirement !

Hilbert's Challenge & Catastrophe

- At the International Conference of Mathematicians in Bologna 1928 set the challenge
 - ★ Given a formal system where Frege's first-order logic is applied to a system of axioms for the natural numbers – Peano arithmetic (PA).
 - ★ Prove that PA is complete.
- Two years later **Kurt Gödel** solved this problem and his resolution was not what Hilbert expected.
- Announced his results at conference in Königsberg in honour of Hilbert's retirement !

Hilbert's program could not succeed.

Gödel upsets the applecart

- **Born:** Brno, Austria–Hungary, April 1906.
- **Died:** Princeton, USA, Jan 1978.
- **Father:** Manager of a textile factory.



- **Born:** Brno, Austria–Hungary, April 1906.
- **Died:** Princeton, USA, Jan 1978.
- **Father:** Manager of a textile factory.



- **Education:**

- ★ University of Vienna, 1924–1930
- ★ Participated in the Vienna Circle, 1926–
- ★ Awarded PhD in Mathematics in 1930

- **Employment:**

- ★ Privatdozent, University of Vienna, 1933–38
- ★ Institute of Advanced Study, Princeton
 - ★ Visitor during, 1933, 1934, 1937
 - ★ Member, 1940–1946
 - ★ Permanent member, 1946–1953
 - ★ Professor, 1953–1976
 - ★ Professor Emeritus, 1976–1978



Kurt Gödel: Major Research Achievements

- **Gödel's incompleteness theorems.**

For any self-consistent recursive axiomatic system powerful enough to describe the arithmetic of the natural numbers, there are true propositions about the natural numbers that cannot be proved from the axioms.

- Showed that neither **the axiom of choice** nor **the continuum hypothesis** can be disproved from the accepted **axioms of set theory**, assuming these axioms are consistent.
- Made important contributions to proof theory by clarifying the connections between classical, intuitionistic, and modal logic.



Kurt Gödel: Fame & fortune during lifetime

- Considered one of the most significant logicians in human history.

“Kurt Gödel's achievement in modern logic is singular and monumental - indeed it is more than a monument, it is a landmark which will remain visible far in space and time. . . . The subject of logic has certainly completely changed its nature and possibilities with Gödel's achievement.” – John von Neumann

- Escaped with wife from Nazi Austria made possible because of mathematical talents.

Works of Gödel especially relevant to CS

- Completeness of first-order logic (1930)
- Incompleteness of formal number theory (1931)
- Papers on decision problems (1932-33)
- Definition of the notion of general recursive function. (IAS lectures, 1934; first published 1965)

- In doctoral dissertation stated the completeness of first-order logic in the form:

“Every valid formula of first-order logic is provable from the logical axioms.”

(Logical axioms refer to those of Frege-Russell-Hilbert.)

- Some definitions:

- ★ Logicians call a mathematical universe that satisfies the axioms a “model” of the axioms.
- ★ a valid formula is a proposition which is true for all models of the axioms.

Example: The statement $S = \text{“}\exists a \text{ with } a * a = 2\text{”}$ is true for \mathbb{R} but false for $\mathbb{N} \implies S$ is not a valid formula if both \mathbb{R} and \mathbb{N} satisfy the axioms of the system.

The Strong Completeness Theorem

- The main focus of Gödel's 1930 paper is

a formula of first-order logic is valid



it is derivable using the rules of inference given earlier by Hilbert and Ackermann.

- But he actually proved more:

Strong Completeness

For any set Σ of first-order axioms

ϕ holds in every model of $\Sigma \iff \phi$ is provable from Σ

- Strong completeness theorem \implies
the rules of inference developed prior to Gödel's work are adequate for deriving all logical consequences of a set of axioms.
- A computer incorporating just those rules will be able to carry out all such derivations.

- **Completeness theorem** implies

Statements provable in any first-order axiomatization of number theory are those that are true in **all** models of the axioms.

- But for **no** recursive axiomatization will those statements coincide with the statements that are true of the natural numbers:

The Gödel-Rosser Theorem

Any recursive axiomatization A of arithmetic must either be inconsistent or else **fail** to prove **both** some statement ϕ **and** its negation. Hence if A is consistent, it must fail to prove some statement that is true of the natural numbers.

Peano's axioms

Here's an informal version of Peano's axioms:

- 0 is a natural number
- Every natural number n has a successor $s(n)$, which is also a natural number. (You can think of the successor of a number n as $n+1$.)
- For every natural number n the successor $s(n)$ is not equal to 0.
- If for any two natural numbers m and n we have $s(m)=s(n)$, then $m=n$.
- The final axiom concerns the method of induction, described at the end of this article.

- Part 1

- ★ Each formula of the theory is assigned a number – Gödel number – ensuring the formula can be recovered from the number.

Example: Consider the premise: “*Anyone in love is happy*” and its symbolic representation

$$(\forall x) ((\exists y) L(x, y) \supset H(x))$$

Can use a simple coding scheme in which each symbol is replaced by

,	L	H	\supset	\forall	\exists	x	y	()
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	1	2	3	4	5	6	7	8	9

Replacing the symbols get

$$846988579186079328699$$

- ★ Numbering extends to cover finite sequences of formulas.

- **Part 2**

★ Formula $PF(x, y)$ is constructed such that

for any two numbers n and m , $PF(n, m)$ holds



n represents a sequence of formulas that constitutes a proof of the formula that m represents

- **Part 3**

- ★ Construct a self-referential formula

“statement S is provable in the system”

- ★ Prove that this sentence is neither provable nor disprovable within the theory.

Larger significance of Gödel's incompleteness paper

- For computer science, Gödel's proof of his incompleteness results more important than the theorems themselves.
- Three aspects of the proof were of particular significance:
 - ★ His precise definition of the class of *primitive recursive functions*.
 - ★ His distinction between object language and meta-language
 - ★ His idea of representing one data type (sequences of strings) by another type (numbers), thereby coding meta-theoretical notions as number-theoretic predicates.

Gödel a computer programmer?

- Martin Davis remarked, the overall structure of Gödel's proof
"looks very much like a computer program"
to anyone acquainted with modern programming languages
- Unsurprisingly, since though *"an actual ... general-purpose ... programmable computer was still decades in the future,"* Gödel faced *"many of the same issues that those designing programming languages and ... writing programs in those languages"* face today.

1. $x/y \equiv (Ez) [z \leq x \ \& \ x = y \cdot z]^{22)}$
 x ist teilbar durch $y^{24)}$.
2. $\text{Prim}(x) \equiv (\overline{Ez}) [z \leq x \ \& \ z \neq 1 \ \& \ z \neq x/x] \ \& \ x > 1$
 x ist Primzahl.
3. $0 \text{ Pr } x \equiv 0$
 $(n+1) \text{ Pr } x \equiv z y [y \leq x \ \& \ \text{Prim}(y) \ \& \ x/y \ \& \ y > n \text{ Pr } x]$
 $n \text{ Pr } x$ ist die n -te (der Größe nach) in x enthaltene Primzahl^{24a)}.
4. $0! \equiv 1$
 $(n+1)! \equiv (n+1) \cdot n!$
5. $\text{Pr}(0) \equiv 0$
 $\text{Pr}(n+1) \equiv z y [y \leq |\text{Pr}(n)|! + 1 \ \& \ \text{Prim}(y) \ \& \ y > \text{Pr}(n)]$
 $\text{Pr}(n)$ ist die n -te Primzahl (der Größe nach).
6. $n \text{ Gli } x \equiv z y [y \leq x \ \& \ x/(n \text{ Pr } x)^y \ \& \ \overline{y}/(n \text{ Pr } x)^{y+1}]$
 $n \text{ Gli } x$ ist das n -te Glied der Zahl x zugeordneten Zahlenreihe (für $n > 0$ und n nicht größer als die Länge dieser Reihe).
7. $l(x) \equiv z y [y \leq x \ \& \ y \text{ Pr } x > 0 \ \& \ (y+1) \text{ Pr } x = 0]$
 $l(x)$ ist die Länge der x zugeordneten Zahlenreihe.
8. $x * y \equiv z z [z \leq |\text{Pr}(l(x) + l(y))|^{24b)} \ \& \ (n) [n \leq l(x) \rightarrow n \text{ Gli } z = n \text{ Gli } x \ \& \ (n) [0 < n \leq l(y) \rightarrow (n+1) \text{ Gli } z = n \text{ Gli } y]]$
 $x * y$ entspricht der Operation des „Aneinanderfügens“ zweier endlicher Zahlenreihen.
9. $R(x) \equiv 2^x$
 $R(x)$ entspricht der nur aus der Zahl x bestehenden Zahlenreihe (für $x > 0$).
10. $E(x) \equiv R(11) * x * R(13)$
 $E(x)$ entspricht der Operation des „Einklammers“ [11 und 13 sind den Grundzeichen „{“ und „}“ zugeordnet].
11. $n \text{ Var } x \equiv (Ez) [13 < z \leq x \ \& \ \text{Prim}(z) \ \& \ x = z^n] \ \& \ n \neq 0$
 x ist eine Variable n -ten Typs.
12. $\text{Var}(x) \equiv (E n) [n \leq x \ \& \ n \text{ Var } x]$
 x ist eine Variable.
13. $\text{Neg}(x) \equiv R(b) * E(x)$
 $\text{Neg}(x)$ ist die Negation von x .

²²⁾ Das Zeichen \equiv wird im Sinne von „Definitionsgleichheit“ verwendet, vertritt also bei Definitionen entweder „=“ oder „ \equiv “ (im übrigen ist die Symbolik die übliche).

²³⁾ Überall, wo in den folgenden Definitionen eines der Zeichen (x) , (Ex) , $z x$ auftritt, ist es von einer Abschätzung für x gefolgt. Diese Abschätzung dient lediglich dazu, um die rekursive Natur des definierten Begriffs (vgl. Satz IV) zu sichern. Dagegen würde sich der Umfang der definierten Begriffe durch Weglassung dieser Abschätzung meistens nicht ändern.

^{24a)} Für $0 < n \leq n$, wenn x die Anzahl der verschiedenen in x enthaltenen Primzahlen ist. Man beachte, daß für $n = x+1$ $n \text{ Pr } x = 0$ ist!

14. $x \text{ Dis } y \equiv E(x) * R(7) * E(y)$
 $x \text{ Dis } y$ ist die Disjunktion aus x und y .
15. $x \text{ Gen } y \equiv E(x) * R(9) * E(y)$
 $x \text{ Gen } y$ ist die Generalisation von y mittels der Variablen x (vorgesetzt, daß x eine Variable ist).
16. $0 \text{ N } x \equiv x$
 $(n+1) \text{ N } x \equiv R(3) * n \text{ N } x$
 $n \text{ N } x$ entspricht der Operation: „ n -maliges Vorsezen des Zeichens „ \wedge “ vor x “.
17. $Z(n) \equiv n \text{ N}[R(1)]$
 $Z(n)$ ist das Zahlzeichen für die Zahl n .
18. $\text{Typ}'_n(x) \equiv (E m, n) [m, n \leq x \ \& \ [m = 1 \vee 1 \text{ Var } m] \ \& \ x = n \text{ N}[R(m)]]^{24b)}$
 x ist Zeichen ersten Typs.
19. $\text{Typ}_n(x) \equiv [n = 1 \ \& \ \text{Typ}'_1(x)] \vee [n > 1 \ \& \ (E v) [v \leq x \ \& \ n \text{ Var } v \ \& \ x = R(v)]]$
 x ist Zeichen n -ten Typs.
20. $\text{Elf}(x) \equiv (E y, z, n) [y, z, n \leq x \ \& \ \text{Typ}_n(y) \ \& \ \text{Typ}_{n+1}(z) \ \& \ x = z * E(y)]$
 x ist Elementarformel.
21. $\text{Op}(x y z) \equiv x = \text{Neg}(y) \vee x = y \text{ Dis } z \vee (E v) [v \leq x \ \& \ \text{Var}(v) \ \& \ x = v \text{ Gen } y]$
22. $\text{FR}(x) \equiv (n) [0 < n \leq l(x) \rightarrow \text{Elf}(n \text{ Gli } x) \vee (E p, q) [0 < p, q < n \ \& \ \text{Op}(n \text{ Gli } x, p \text{ Gli } x, q \text{ Gli } x)] \ \& \ l(x) > 0]$
 x ist eine Reihe von Formeln, deren jede entweder Elementarformel ist oder aus den vorhergehenden durch die Operationen der Negation, Disjunktion, Generalisation hervorgeht.
23. $\text{Form}(x) \equiv (E n) [n \leq |\text{Pr}(l(x))| * \text{Op}(x)] \ \& \ \text{FR}(n) \ \& \ x = [l(n)] \text{ Gli } n^{25)}$
 x ist Formel (d. h. letztes Glied einer Formelreihe n).
24. $e \text{ Geb } n, x \equiv \text{Var}(x) \ \& \ \text{Form}(x) \ \& \ (E a, b, c) [a, b, c \leq x \ \& \ x = a * (v \text{ Gen } b) * c \ \& \ \text{Form}(b) \ \& \ l(a) + 1 \leq n \leq l(a) + l(v \text{ Gen } b)]$
 Die Variable v ist in x an n -ter Stelle gebunden.

^{24b)} $m, n \leq x$ steht für: $m \leq x \ \& \ n \leq x$ (ebenso für mehr als 2 Variable).

²⁵⁾ Die Abschätzung $n \leq |\text{Pr}(l(x))|^{25a)}$ erkennt man etwa so: Die Länge der kürzesten von x gebildeten Formelreihe kann höchstens $l(x)$ Teillformeln Anzahl der Teillformeln von x sein. Es läßt aber höchstens $l(x)$ Teillformeln der Länge 1, höchstens $l(x)-1$ der Länge 2 usw., in genau also höchstens $\frac{l(x)[l(x)+1]}{2} \leq l(x)^2$. Die Primzahlen aus n können also sämtlich kleiner als $\text{Pr}([l(x)]^2)$ angenommen werden, ihre Anzahl $\leq l(x)^2$ und ihre Exponenten (welche Teillformeln von x sind) $\leq x$.

Killer blow to Hilbert's Program

- In Sept 1930 there was a conference in Königsberg and the key logicians were present.
- The conference also included Hilbert's retirement address from the University of Göttingen.
- Gödel announced his incompleteness theorem at a round-table discussion session on the day 3 of the conference.
- Announcement drew little attention except from von Neumann who instantly grasped the importance of Gödel's theorem.
- Following from the incompleteness, von Neumann realised consistency itself is unprovable and concluded that was the end of Hilbert's program.

- After the conference Gödel published

Second Incompleteness Theorem

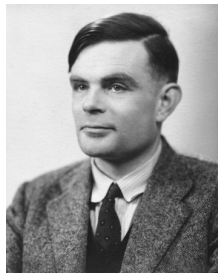
No reasonable, consistent mathematical system can prove its own consistency.

Philosophical significance of the incompleteness theorems

- Much later, in his Gibbs Lecture to the American Mathematical Society (1951)
- Gödel would suggest that the incompleteness theorems are relevant to the questions
 - ★ whether the powers of the human mind exceed those of any machine,
 - ★ and whether there are mathematical problems that are undecidable for the human mind.

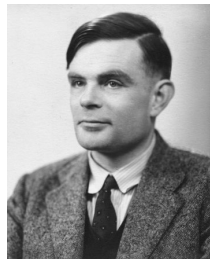
**Turing conceives of
the all-purpose computer**

- **Born:** London, England, June 1912.
- **Died:** Wilmslow, England, June 1954.
- **Father:** Worked for the Indian Civil Service.



- **Education:**

- ★ University of Cambridge, 1931–1934
- ★ BA in mathematics 1934
- ★ Princeton University, 1936–1938
- ★ Awarded PhD 1938



- **Employment:**

- ★ Fellow of King's College, Cambridge, 1935–39
- ★ Government Code and Cypher School, Bletchley Park, 1939 – 1945
- ★ National Physical Laboratory, London, 1945-1947
- ★ Reader in Mathematics, University of Manchester, 1948–1954



Alan Turing: Major Research Achievements

- Proved there is no solution to the *Entscheidungsproblem*.
- Formalized the concept of *algorithm* and *computation* with the Turing machine.
- Created one of the first designs for a stored-program computer (ACE).
- Father of computer science and artificial intelligence.
- Turing test.
- Invented LU matrix decomposition method !



- Fellow of the Royal Society
- Awarded OBE (Officer of the Order of the British Empire) for his wartime efforts.
- His wartime work was subject to the Official Secrets Act during his lifetime, so was not common knowledge.
- Criminally prosecuted for homosexual acts near the end of his life and had to accept treatment with female hormones as an alternative to prison

- Leibniz's dream
 - ★ human reason reduced to calculation **and**
 - ★ of powerful computers to carry out these calculations.
- Frege provided system of rules to account for human deductive reasoning.
- Gödel had proved Frege's rules were complete.
- Time to attack Hilbert's Entscheidungsproblem
 - Can it always be determined, by a process, whether a stated conclusion can be derived from a set of premises using Frege's rules of logic?
- An algorithm for Hilbert's *Entscheidungsproblem* would reduce all human deductive reasoning to brute calculation and fulfil of Leibniz's Dream.

Turing's exposure to the Entscheidungsproblem

- Spring of 1935 Turing attended Max Newman's Part III course on the foundations of mathematics.
- Turing learned of the Entscheidungsproblem.
- Convinced its solution depended on the formalisation of the notion of "*process*"

"Oh I knew [Turing] very well. . . . I believe it all started because he attended a lecture of mine on foundations of mathematics and logic in which I had mentioned in the lecture the importance of having such a definition and I think I said, in the course of this lecture, that what is meant by saying that the process is constructive is that it's purely a mechanical machine and I may even have said a machine can do it. But he took the notion and really tried to follow it right up and did produce this extraordinary definition of a perfectly general, what he called, computable function. Thus, giving the first idea really of a perfectly general computing machine." – Max Newman, 1975

Turing's Analysis of the Computation Process

Turing's solution combined the physical and the abstract.

"We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q_1, q_2, \dots, q_R which will be called " m -configurations". The machine is supplied with a "tape", (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r -th, bearing the symbol $S(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its m -configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the m -configuration q_n and the scanned symbol $S(r)$. This pair $q_n, S(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the m -configuration may be changed. Some of the symbols written down will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure" – (A. M. Turing, 1936 pp.231-2)

- Elements of a Turing machine:
 - ★ A list of all possible states
 - ★ A list of all symbols
 - ★ For each state it must be stated what action to perform when a particular symbol is encountered on the tape.
- An action consists of possibly
 - ★ changing the symbol on the square being scanned,
 - ★ moving one square to the left or to the right,
 - ★ a change of state.

- The formula $R a : b \rightarrow S$ symbolizes the action:

When the machine is in state R scanning the symbol a on the tape, it will

- ★ replace a by b ,
- ★ move one square to the right and
- ★ then shift into state S .

- The formula $R a : b \leftarrow S$ symbolizes the action:

When the machine is in state R scanning the symbol a on the tape, it will

- ★ replace a by b ,
- ★ move one square to the **left** and
- ★ then shift into state S .

- The formula $R a : b \star S$ symbolizes the action:

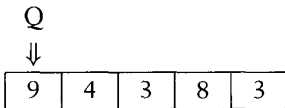
When the machine is in state R scanning the symbol a on the tape, it will

- ★ replace a by b ,
- ★ then shift into state S .

Consider this Turing Machine

Q 0 : $\square \rightarrow E$	Q 2 : $\square \rightarrow E$	Q 4 : $\square \rightarrow E$	Q 6 : $\square \rightarrow E$	Q 8 : $\square \rightarrow E$
Q 1 : $\square \rightarrow O$	Q 3 : $\square \rightarrow O$	Q 5 : $\square \rightarrow O$	Q 7 : $\square \rightarrow O$	Q 9 : $\square \rightarrow O$
E 0 : $\square \rightarrow E$	E 2 : $\square \rightarrow E$	E 4 : $\square \rightarrow E$	E 6 : $\square \rightarrow E$	E 8 : $\square \rightarrow E$
E 1 : $\square \rightarrow O$	E 3 : $\square \rightarrow O$	E 5 : $\square \rightarrow O$	E 7 : $\square \rightarrow O$	E 9 : $\square \rightarrow O$
O 0 : $\square \rightarrow E$	O 2 : $\square \rightarrow E$	O 4 : $\square \rightarrow E$	O 6 : $\square \rightarrow E$	O 8 : $\square \rightarrow E$
O 1 : $\square \rightarrow O$	O 3 : $\square \rightarrow O$	O 5 : $\square \rightarrow O$	O 7 : $\square \rightarrow O$	O 9 : $\square \rightarrow O$
E \square : 0 * F	O \square : 1 * F			

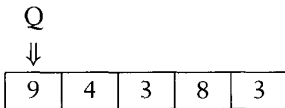
and applying it to this input



Consider this Turing Machine

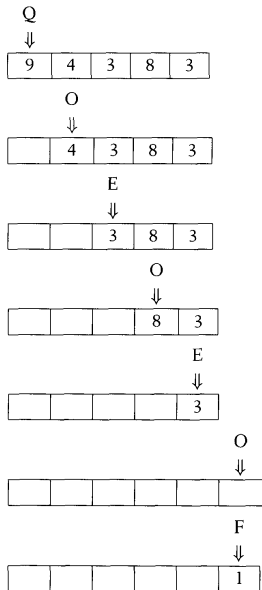
Q 0 : □ → E	Q 2 : □ → E	Q 4 : □ → E	Q 6 : □ → E	Q 8 : □ → E
Q 1 : □ → O	Q 3 : □ → O	Q 5 : □ → O	Q 7 : □ → O	Q 9 : □ → O
E 0 : □ → E	E 2 : □ → E	E 4 : □ → E	E 6 : □ → E	E 8 : □ → E
E 1 : □ → O	E 3 : □ → O	E 5 : □ → O	E 7 : □ → O	E 9 : □ → O
O 0 : □ → E	O 2 : □ → E	O 4 : □ → E	O 6 : □ → E	O 8 : □ → E
O 1 : □ → O	O 3 : □ → O	O 5 : □ → O	O 7 : □ → O	O 9 : □ → O
E □ : 0 * F	O □ : 1 * F			

and applying it to this input



What output will it produce?

Truing Machines in Action



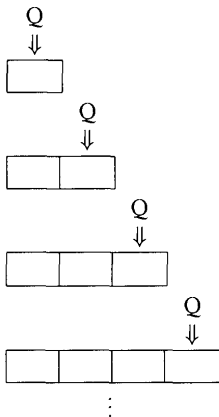
Q 0 : □ → E	Q 2 : □ → E	Q 4 : □ → E	Q 6 : □ → E	Q 8 : □ → E
Q 1 : □ → O	Q 3 : □ → O	Q 5 : □ → O	Q 7 : □ → O	Q 9 : □ → O
E 0 : □ → E	E 2 : □ → E	E 4 : □ → E	E 6 : □ → E	E 8 : □ → E
E 1 : □ → O	E 3 : □ → O	E 5 : □ → O	E 7 : □ → O	E 9 : □ → O
O 0 : □ → E	O 2 : □ → E	O 4 : □ → E	O 6 : □ → E	O 8 : □ → E
O 1 : □ → O	O 3 : □ → O	O 5 : □ → O	O 7 : □ → O	O 9 : □ → O
E □ : 0 * F	O □ : 1 * F			

Turing Machines have no physical limitations

Turing machine consisting of

$$Q \square : \square \rightarrow Q$$

when started on a blank tape will keep on moving right "*forever*"

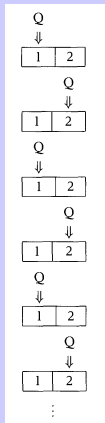


Turing Machines may not halt

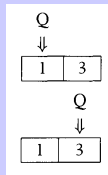
Turing machine

$Q_1 : 1 \rightarrow Q$ and $Q_2 : 1 \leftarrow Q$

Input 12 it will bounce back and forth



while on input 13 it will halt



Coding a Turing machine as a natural number

Probably inspired by Gödel Turing encoded a machine as a natural number.

Example:

- Machine for distinguishing between even and odd numbers

Q 0 : $\square \rightarrow E$	Q 2 : $\square \rightarrow E$	Q 4 : $\square \rightarrow E$	Q 6 : $\square \rightarrow E$	Q 8 : $\square \rightarrow E$
Q 1 : $\square \rightarrow O$	Q 3 : $\square \rightarrow O$	Q 5 : $\square \rightarrow O$	Q 7 : $\square \rightarrow O$	Q 9 : $\square \rightarrow O$
E 0 : $\square \rightarrow E$	E 2 : $\square \rightarrow E$	E 4 : $\square \rightarrow E$	E 6 : $\square \rightarrow E$	E 8 : $\square \rightarrow E$
E 1 : $\square \rightarrow O$	E 3 : $\square \rightarrow O$	E 5 : $\square \rightarrow O$	E 7 : $\square \rightarrow O$	E 9 : $\square \rightarrow O$
O 0 : $\square \rightarrow E$	O 2 : $\square \rightarrow E$	O 4 : $\square \rightarrow E$	O 6 : $\square \rightarrow E$	O 8 : $\square \rightarrow E$
O 1 : $\square \rightarrow O$	O 3 : $\square \rightarrow O$	O 5 : $\square \rightarrow O$	O 7 : $\square \rightarrow O$	O 9 : $\square \rightarrow O$
E \square : 0 * F	O \square : 1 * F			

- List the instructions one after another separated by a semi-colon

$$Q0 : \square \rightarrow E ; Q1 : \square \rightarrow O ; Q2 : \square \rightarrow E ; \dots$$

- Replace each symbol by a string of decimal digits.

Coding a Turing machine as a natural number

Say Q, E, O, F are coded by 99, 919, 929 and 939 and other symbols by

Symbol	Representation	Symbol	Representation
0	8008	□	8558
1	8018	→	616
2	8028	←	626
3	8038	*	636
4	8048	:	646
5	8058	;	77
6	8518		
7	8528		
8	8538		
9	8548		

The Turing machine is then coded by the number (easy to decode number)

```
9980086468558616919 77 9980286468558616919 77 9980486468558616919 77
9985186468558616919 77 9985386468558616919 77 9980186468558616929 77
9980386468558616929 77 9980586468558616929 77 9985286468558616929 77
9985486468558616929 77 91980086468558616919 77 91980286468558616919 77
91980486468558616919 77 91985186468558616919 77 91985386468558616919 77
91980186468558616929 77 91980386468558616929 77 91980586468558616929 77
91985286468558616929 77 91985486468558616929 77 92980086468558616919 77
92980286468558616919 77 92980486468558616919 77 92985186468558616919 77
92985386468558616919 77 92980186468558616929 77 92980386468558616929 77
92980586468558616929 77 92985286468558616929 77 92985486468558616929 77
91985586468008636939 77 92985586468018636939
```


The Halting Set of a Turing Machine

- When a Turing machine is applied to a number it may or may not halt.
- The **halting set** \mathcal{S} of a machine is the set of natural numbers s.t.

if $n \in \mathcal{S}$ then the machine will halt when applied to n

- List all Turing machines and their halting set

\mathcal{S}_1 be the halting set of Turing machine coded by $n_1 \in \mathbb{N}$

\mathcal{S}_2 be the halting set of Turing machine coded by $n_2 \in \mathbb{N}$

\mathcal{S}_3 be the halting set of Turing machine coded by $n_3 \in \mathbb{N}$

⋮

- Can apply the diagonal method to generate a new set \mathcal{S} which is different from all the \mathcal{S}_i 's in the list

Turing applies Cantor's Diagonal Method

- The diagonal method works so that

$$n_i \in \mathcal{S} \text{ if } n_i \notin \mathcal{S}_i$$

while

$$n_i \notin \mathcal{S} \text{ if } n_i \in \mathcal{S}_i$$

- Therefore
 - ★ \mathcal{S} is different from each halting set in the original list
 - ★ \mathcal{S} is not the halting set for any of the Turing machines in the list
- **Thus \mathcal{S} is not the halting set of any Turing machine**

Turing applies Cantor's Diagonal Method

- The diagonal method works so that

$$n_i \in \mathcal{S} \text{ if } n_i \notin \mathcal{S}_i$$

while

$$n_i \notin \mathcal{S} \text{ if } n_i \in \mathcal{S}_i$$

- Therefore
 - ★ \mathcal{S} is different from each halting set in the original list
 - ★ \mathcal{S} is not the halting set for any of the Turing machines in the list
- **Thus \mathcal{S} is not the halting set of any Turing machine**

So what??

Unsolvable problems and *Entscheidungsproblem*

- Solution of the *Entscheidungsproblem* would provide an algorithm to settle all mathematical questions.
- If there is any mathematical problem which is shown to be algorithmically unsolvable then the *Entscheidungsproblem* is unsolvable.

Consider this problem (an example of an unsolvable problem)

Find an algorithm to determine for a given natural number whether it belongs to \mathcal{S}

- If problem solvable $\implies \exists$ a Turing machine able to accomplish task.
- Machine should halt with a blank tape except for one digit
 - ★ **1** if the input number belongs to \mathcal{S}
 - ★ **0** if the input number does not belong to \mathcal{S}
- Also say the machine halts in state F and has no instructions for state F .
- Now add two instructions to the machine

$$F0 : \square \rightarrow F \quad \text{and} \quad F\square : \square \rightarrow F$$

- How does this new machine differ from the original one?
 - ★ ✓ It halts on input numbers from \mathcal{S}
 - ★ ✗ It does not halt on input numbers not from \mathcal{S}
- This new machine has halting set \mathcal{S}
- **Contradiction** as \mathcal{S} is not the halting set of any Turing machine.
- Therefore the premise that the problem

Find an algorithm to determine for a given natural number whether it belongs to \mathcal{S}

is solvable is false.

Therefore \exists a problem not algorithmically solvable \implies
Entscheidungsproblem is unsolvable.

- No Turing machine could solve the *Entscheidungsproblem* problem.
- But perhaps maybe other types of computations were possible?
- Turing showed that a variety of complicated calculations could be done on a Turing machine.
- While testing the validity of what he had done he came up with the idea of the **Universal Turing Machine**

Turing's Universal Machine

- Imagine two natural numbers on a Turing machine \mathcal{M} tape separated by a blank space.
 - ★ First number is the code of some Turing machine.
 - ★ Second is the input to that machine.

Code number of a Turing machine \mathcal{M}		Input to \mathcal{M}
---	--	--

- Say \mathcal{M} is able to decipher and implement the instructions of the machine coded via by the first number via its own instructions.
- \implies this one single machine \mathcal{M} can perform the calculations of any Turing machine.
- This is the Universal Turing Machine and it anticipates modern day computer software *software* and *programming*.

- In April 1936 Turing gave Newman a draft of his answer to the Entscheidungsproblem.
- Soon after Newman had convinced himself of the correctness of Turing's reasoning, he received a copy of Alonzo Church's "*An Unsolvable Problem of Elementary Number Theory*". Church had also proved the undecidability of the Entscheidungsproblem but in a less intuitive and accessible fashion.
- Turing's paper was submitted to the Proceedings of the *London Mathematical Society* in May 1936 and was refereed by Church.
- Newman made arrangements for Turing to attend Princeton with the purpose of working towards a doctoral degree under Church's supervision.

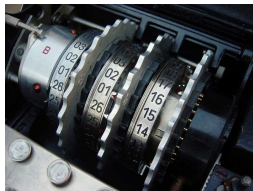
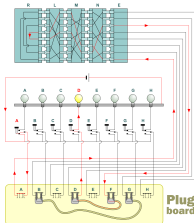
- September 1936 to July 1938 studied under Alonzo Church at Princeton University.
- Obtained PhD from Princeton in June 1938.
- **Thesis:** Hierarchy of systems s.t.
System 1's axioms \subset System 2's axioms \subset System 3's axioms $\subset \dots$
and propositions undecidable in System $i - 1$ are decidable in System i .
- Considered Turing machines augmented by oracles to help categorize the level of *unsolvability* of problems.
- Familiarized himself with available technology – built a device, using electromechanical relays, that multiplied numbers in binary notation.

- At this time Princeton had a high concentration of mathematical talent – Weyl, Einstein, von Neumann, Church,
- Turing would have meet **John von Neumann**. Did they have discussions?
- At this period von Neumann was not working on the foundations of mathematics and logic.
- After Gödel's results von Neumann claimed he never again read a paper in logic.
- However, Turing's work definitely influenced von Neumann's thinking about computers during and after WW II.
- When did von Neumann become aware of Turing's work ???

- Turing returned to Cambridge in 1938.
- Recruited almost immediately to help break the German military codes.
- Arrived in Bletchley Park on Sept 4th 1939.
- Worked on deciphering messages encoded by **Enigma** machine



The Enigma circuit



This required finding the Enigma's settings for each day.

- Designed an electro-mechanical machine *Bombe* to crack Enigma
- Bombe searched for possible correct settings used for an Enigma message (rotor and plugboard settings) from a suitable **crib**:

Encrypted text

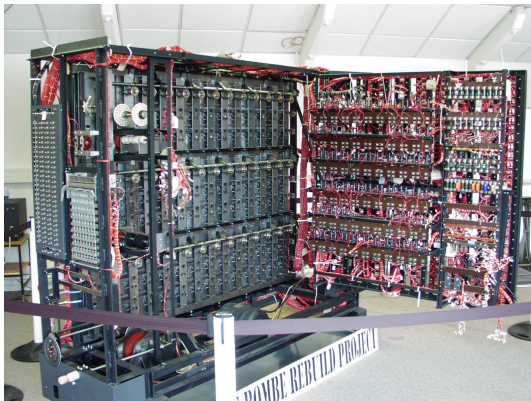
p	e	g	m	u	o	x	y	q	p	w	t	j	a	b	x	l	p	v
w	e	t	t	e	r	v	o	r	h	e	r	s	a	g	e			

Encrypted text

p	e	g	m	u	o	x	y	q	p	w	t	j	a	b	x	l	p	v
→	w	e	t	t	e	r	v	o	r	h	e	r	s	a	g	e		

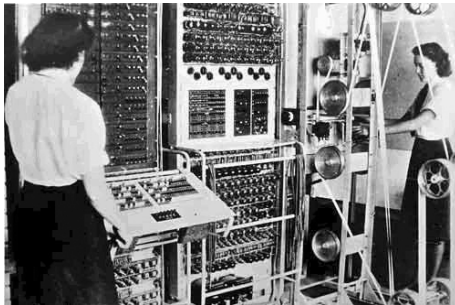
- For a possible setting of the rotors Bombe performed a chain of logical deductions based on the crib, implemented electrically.
- The Bombe detected when a contradiction occurred, ruled out that setting, and moved on to the next.
- Most of the possible settings would cause contradictions and be discarded, leaving only a few to be investigated in detail.

- The first bombe was installed on 18 March 1940.



- More than two hundred Bombes were in operation by the end of the war – all of them were destroyed at the end of the war.

- **July 1942:** Turing devised a method to decypher *Lorenz cipher messages* produced by the Germans' *Geheimschreiber* machine.
- Tommy Flowers and Max Newman, went on to build **Colossus** (1943-44), an embodiment of Turing's method, the world's first automatic calculation device. – vacuum tube circuits were used.

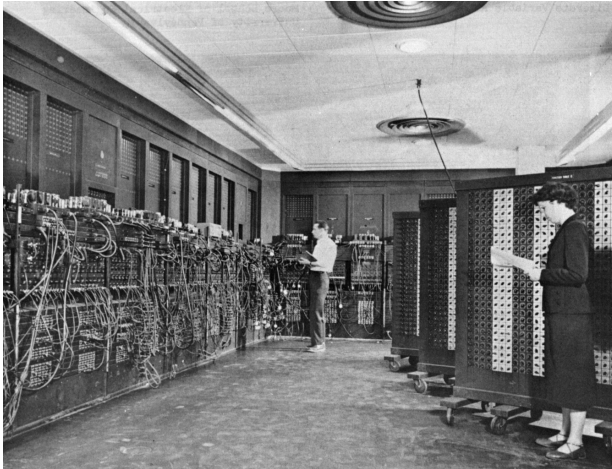


Building the first Universal Computer

John von Neumann and the Moore School

- Von Neumann (1903-57) worked on Hilbert's program until Gödel's incompleteness theorems.
- Moved to Princeton (1930) and vowed never to read a paper on logic again !
- During WW II was asked to participate in the project
to construct a powerful electronic calculator, the ENIAC
at the *Moore School of Electrical Engineering* in the *UPenn*.
- ENIAC had 18,000 vacuum tubes compared to Colossus 1,500.
- ENIAC's chief engineer was John Presper Eckert, Jr.
- ENIAC though a digital device, its components were built to be functionally similar to those in differential analyzers.

The ENIAC



John von Neumann and the EDVAC draft report

- When von Neumann joined the Moore school the ENIAC was more or less up and running.
- Turned his attentions to the next planned computer EDVAC.
- In June 1945 produced his famous "*First Draft of a Report on the EDVAC*"
- It described a computer
 - ★ where arithmetic operations performed in binary and
 - ★ containing a component which brings instructions to be executed one at a time from the memory into the arithmetic component.
- This way to organize a computer known as the **von Neumann architecture**.

- Report emphasized the EDVAC should be all-purpose...
- and “*logical control*” of a computer is crucial for its being “*as nearly as possible all-purpose*”.
- Strong echoes of Turing in the work but no references made.
- Von Neumann showed EDVAC’s general applicability by programming it to simply sort data efficiently.
- Much to Eckert and Muachly’s annoyance the report only had von Neumann’s name on it !
- Unclear how much of the EDVAC report represent von Neuman’s personal contribution?

Post-war computers required memory

- Post-war computers designed to be all-purpose universal devices.
- Capable of carrying out any symbolic process as long as the steps in process were sufficiently precise.
- Therefore needed the **stored program concept**.
- ... and memory which had to be
 - ★ large to store these "*instructions*" and data. - didn't have Turing's infinite length tape !
 - ★ sufficiently fast so data or instructions could be accessed in a single step i.e. random access. - no shunting along one square of tape at a time!
- 1940s, two candidate devices: *mercury delay line* (EDVAC solution) and *cathode ray tube* (Manchester University solution).



Finished 1949, operational 1951

- Completion of EDVAC was delayed.
- Eckert and Mauchly wanted to commercialize their work.
- Became embroiled in patent disputes with UPenn. awarded patent for ENIAC, but not for EDVAC
- They departed UPenn to form the *Eckert-Mauchly Computer Corporation*

- After the war Turing joined the National Physics Lab.
- According to his mother, Turing was concerned with *“his plans for the construction of a universal computer and of the service such a machine might render to psychology in the study of the human brain”* – E. S. Turing, 1959
- His ideas came together in his report ACE Report of 1945 –
pre-empted to some extent by the von Neumann's draft EDVAC report
- However, went beyond von Neumann's scope and conception.
- Turing's ACE not restricted to routine calculation. Some of the problems within its repertoire:
 - ★ The solution of simultaneous linear equations
 - ★ Finding the solution for a simple jigsaw problem.
 - ★ From a given Chess position, to calculate all the winning positions for three moves on either side.

- ACE was designed in a minimal way.
- Many operations were to be carried out by programming.
- When a proposal was made to modify ACE in a von Neumann direction he responded:

“It is . . . very contrary to the line of development here, and much more in the American tradition of solving one’s difficulties by means of much equipment than by thought. . . Furthermore certain operations which we regard as more fundamental than addition and multiplication have been omitted.” – Alan Turing, 1959

- Unfortunately, NPL did not prioritize building ACE. In disillusionment Turing left in 1947 for Manchester University.
- However, the Pilot Model ACE – a less ambitious version of ACE – was eventually built and first ran in 1950.

Stored memory: Whose idea ?

- **Version 1**

Product of von Neumann's genius alone and the EDVAC draft report is evidence of this.

“ Von Neumann was the first person, as far as I am concerned, who understood explicitly that a computer is essentially performed logical functions, and that the electrical aspects were ancillary.” – Goldstine, *The computer from Pascal to von*

Neumann, 1972

- **Version 2**

EDVAC report was a reflection of the joint thinking of the Moore school group at UPenn \implies Eckert a vital contributor.

- **Version 3**

Turing, though unattributed, influenced and inspired von Neumann.

- Eckert probably did not envision a *universal computer*. Why?
 - ★ Design of the ENIAC is very far from a universal computer.
 - ★ It contains modules replicating components of the analogue differential analyzers which are unnecessary for a digital computer.
 - ★ Though he proposed the mercury delay line as a solution to EDVAC's memory problem, in a memo spoke of automatic programming set up on alloy discs.
- **Version 3** is the most plausible and believes it is now the most common view.

“Virtually all computers today from \$10 million supercomputers to the tiny chips that power cell phones and Furbies, have one thing in common: they are all “von Neumann machines”, variations on the basic computer architecture that John von Neumann, building on the work of Alan Turing, laid out in the 1940s” – Time magazine, March 19, 1999