## Alan Turing on Computability and Intelligence

## Computer Science and Philosophy, TT 2019

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy
Hertford College, Oxford


1

## Overall Plan of the Lectures

```
- Lecture 1: Types of number, Cantor, infinities, diagonal arguments
■ Lecture 2: Hilbert's Programme, Gödel's Theorem
- Lectures 3-5: The 1936 paper, "On Computable Numbers, with an Application to the Entscheidungsproblem", which introduced the Turing Machine.
- Lectures 6-8: The 1950 paper, "Computing Machinery and Intelligence", which introduced the Turing Test.
3
```

3

Highly Recommended Reading

- Andrew Hodges, Alan Turing: The Enigma of Intelligence (Hutchinson, 1983, 586 pp.)
An excellent biography by a Fellow of Wadham.
- Andrew Hodges, Turing: A Natural Philosopher (Phoenix, 1997, 58 pp.)
Well worth reading for a quick overview and insights into Turing's life and thought.
- S. Barry Cooper and Jan van Leeuwen (eds), Alan Turing: His Work and Impact (Elsevier, 2013, 914 pp.) Very rich collection of Turing's work with commentary from philosophers, mathematicians, computer scientists etc.
- Ernest Nagel and James R. Newman, Gödel's Proof (RKP, 1959 \& 1971, 118 pages)


## Role of the Course

- This course is part of the "Introduction to Philosophy" for Prelims in Computer Science and Philosophy.
- The 3-hour examination will contain six "either ... or" questions on "General Philosophy" and six questions on Turing.
- Students must answer four questions, including at least one from each half of the paper (so between one and three on Turing).

2

## Supporting Textbook

- Charles Petzold, The Annotated Turing, contains "everything you need to know" on the background and the classic 1936 paper.
- It reproduces the original paper, together with extensive explanation and discussion.
- With its help, you should aim to read and understand Turing's entire paper.

4

4

- Turing was born in London on June 23, 1912, and educated at Hazelhurst and Sherborne Schools (while his parents lived in India, his father working in the Indian Civil Service).
- He was shy, but formed a close relationship with Christopher Morcom. In 1929, both applied to read Mathematics at Trinity College, Cambridge: Morcom was accepted, but Turing was not.
- Morcom seems to have been Turing's first love; he tragically died of tuberculosis (in February 1930), with profound effects on Turing.

6

## Turing on Computability and Intelligence

E (Later in his life, despite his heroic - but largely unknown - decryption of Nazi Enigma codes during the War, Turing was prosecuted for homosexuality, subsequently killing himself in 1954.)

- Turing reapplied to Cambridge in 1930, and was accepted for King's College, going up in 1931.
E In May/June 1934, Turing took his Part II Tripos examinations, passing with Distinction and winning a King's College research studentship. In 1935, he was elected to a 3-year fellowship.
- In Spring 1935, Turing attended a Part III course by Max Newman, on the Foundations of Mathematics, largely inspired by David Hilbert ...

7

## Hilbert's Tenth Problem

E Hilbert's list later grew to 23, including:
10. Determination of the Solvability of a Diophantine Equation
Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.

* For lucid discussion of this problem and the history of its (dis-) solution, see Petzold, chapters 1 and 18.

9

## Some of Hilbert's Other Problems

E 1. The continuum hypothesis: $\aleph_{1}=2 \aleph_{0}$ ? This asks whether the second transfinite number is the cardinality of the continuum - see Petzold p. 32. It has since been shown to be neither provable nor disprovable within standard set theory.

- 2. Prove the consistency of arithmetic axioms; See Petzold p. 46, and discussion below.
E 8. Riemann hypothesis and other problems including Goldbach's Conjecture;
Goldbach is mentioned by Petzold on p. 47.


## Diophantine Equations

These are named after problems discussed by the $3^{\text {rd }}$ century Alexandrian mathematician Diophantus, often called "the father of algebra" though some of his methods were actually Babylonian.

- Diophantine equations are polynomial equations (which may have many, or no solutions) whose solutions must be integers. The most famous are associated with Fermat's "Last Theorem": $x^{n}+y^{n}=z^{n}$
Fermat claimed to have a proof that there are no integer solutions if $n>2$. This "theorem" was finally proved by Andrew Wiles (Merton College) in 1995.

10

## Hilbert's Formalist Challenges

In 1921 Hilbert set out his famous "programme" in the foundations of mathematics, later extended:

- 1921: To establish mathematics on a solid and provably consistent foundation of axioms, from which, in principle, all mathematical truths can be deduced (by the standard rules of first order predicate logic).
- 1928: the Entscheidungsproblem or "decision problem": can an effective procedure be devised which would demonstrate - in a finite time - whether any given mathematical proposition is, or is not, provable from a given set of axioms?


## Turing on Computability and Intelligence

Turing's 1936 Paper:
"On Computable Numbers, with an Application to the Entscheidungsproblem"

- Turing's classic paper, which introduced the Turing Machine, is the main focus of these lectures. It settled Hilbert's decision problem (by showing that it cannot be solved).
- But the paper starts from the concept of a "computable number", and in order to put this in context, we must first understand several other types of number ...


## Various Types of Infinity

- Georg Cantor (1845-1918) developed set theory and the theory of transfinite cardinal numbers.
- His work implies that we must recognise more than one "infinite number".

- The key to understanding such numbers is widely known as "Hume's Principle" ...

15

## One-to-one Correspondence

E Making Hume's principle a bit more precise, we shall say that two sets have the same cardinality if, and only if, a bijective function (or bijection, or one-to-one correspondence) can be defined between them.

- Such a function is both one-to-one (or injective: distinct elements of the domain are never mapped to the same element of the codomain) and onto (or surjective: the range/image of the function is the entire codomain).


## Various Types of Number

- All of the following sets of numbers are standardly used in mathematics:
- Natural numbers 1, 2, 3, 4, .. (set $\mathbb{N}$ ), which is a subset of the set of integers (set $\mathbb{Z}$ );
- Rational numbers, fractions of integers (set $\mathbb{Q}$ );
- Real numbers (set $\mathbb{R}$ ), which include also:
- Irrational numbers, which are not rational;
- Algebraic numbers, roots of algebraic equations;
- Transcendental numbers, which are not algebraic;
- Complex numbers (set $\mathbb{C}$ ) need not concern us.

14
14

## Frege cites "Hume's Principle"

"... When we have ... acquired a means of arriving at a determinate number and of recognizing it again as the same, we can assign it a number word as its proper name.
§ 63. HUME long ago mentioned such a means: 'When two numbers are so combined as that the one has always an unit answering to every unit of the other, we pronounce them equal.' ..."

Frege, Foundations of Arithmetic
(1884, translated by J. L. Austin 1950),
16 referring to Hume's Treatise 1.3.1.5

16


18

## Making Do With Surjectivity

- When dealing with infinite sets, rather than attempting to define a bijection, we often make do with the principle that if we can define a surjective function:

$$
F: \mathrm{A} \rightarrow \mathrm{~B}
$$

then A must be "at least as big" as $B$; or in terms of cardinalities, $|A| \geq|B|$. (Intuitively, $A$ must provide enough "arrows" to hit all of B.)

- Surjections in both directions would imply
$(|A| \geq|B|) \wedge(|B| \geq|A|) \quad \therefore|A|=|B|$

19

## Rational Numbers

E A rational number is a number that can be expressed as a fraction of integers.

E Expressed as a decimal, a number is rational if and only if that decimal eventually recurs (i.e. repeats infinitely), e.g. 3.000... or $7.5000 \ldots$ or $0.666 \ldots$ or $0.367367367 \ldots$

- To convert to a fraction, sum the series:
$0.367367367 \ldots=\frac{367}{1000}+\frac{367}{1000000}+\frac{367}{1000000000}+\cdots$
$=\frac{367}{1000} \times \frac{1000}{999}$ [using the formula " $a /(1-r)$ "]

21

- We thus list all of the fractions in the array, starting $1 / 1,2 / 1,1 / 2,1 / 3,2 / 2,3 / 1,4 / 1 \ldots$
- So our surjection maps 1 to the $1^{\text {st }}$ of these, 2 to the $2^{\text {nd }}, 3$ to the $3^{\text {rd }}$, and so on.
- Curiously, every fraction (including all the natural numbers $1 / 1,2 / 1$ etc.) will appear in this list an infinite number of times, e.g. $1 / 3$ will appear again as $2 / 6,3 / 9,4 / 12$ etc.
- If we want to include negative fractions, we can alternate: $1 / 1,-1 / 1,2 / 1,-2 / 1,1 / 2$ etc. (and we could start with 0 if desired).


## Enumerability / Countability

E If a surjection can be defined from the set of natural numbers $\mathbb{N}$ to set $A$, then we say that A is countable or enumerable (i.e. they can in principle be enumerated, or put in a list).

- Here, following Petzold (and tradition), we take the natural numbers to start from 1 (rather than from 0, as many authors do).
- Georg Cantor proved that the set of rational numbers $\mathbb{Q}$ and the set of algebraic numbers are both enumerable.

20

## Rational Numbers are Enumerable

- Set out the fractions of positive integers in a grid, and define a surjection $F: \mathbb{N} \rightarrow \mathbb{Q}$ following the line: $F(1)=1 / 1, F(2)=2 / 1, F(3)=1 / 2$, etc.


22
> $\sqrt{ } 2$ is Not Rational
> E Suppose that $\sqrt{ } 2=\frac{m}{n}$, where $m$ and $n$ are integers with no common factors.
> - Squaring, $2=\frac{m^{2}}{n^{2}}$, so $2 n^{2}=m^{2}$, and $m^{2}$ is even.

> But any odd number (e.g. $2 k+1$ ) has an odd square $\left(4 k^{2}+4 k+1\right)$, so $m$ cannot be odd, i.e. $m$ is even.
> - Therefore $m=2 k$, for some integer $k$.
> - Hence $2 n^{2}=m^{2}=4 k^{2}$, and $n^{2}$ is even.
> - So $n$ is even too, meaning that $m$ and $n$ have the factor 2 in common ... CONTRADICTION!!

## Turing on Computability and Intelligence

## Algebraic Numbers

- An algebraic number is any number that can be a solution (or "root") of an equation in one variable with integer coefficients (these are called algebraic equations).
- Any rational number is straightforwardly algebraic, since ${ }^{a} / b$ ( $a$ \& $b$ integers) is the solution to the equation $b x=a$.
- Any square root or $n^{\text {th }}$ root of a rational number is algebraic, being a solution to some equation of the form $b x^{n}=a$.

So now we can put all possible algebraic equations in order, starting like this:

## Rank 2:

$-x+0=0, \quad x+0=0$

## Rank 3:

$-2 x+0=0, \quad-x-1=0, \quad-x+1=0$,
$x-1=0, \quad x+1=0, \quad 2 x+0=0$,
$-x^{2}+0=0, \quad x^{2}+0=0$
Rank 4:
$-3 x+0=0, \quad-2 x-1=0, \quad-2 x+1=0$,
27
27

## Transcendental Numbers

- A transcendental number is a number which is not algebraic - so it is not a solution to any algebraic equation.
- $\pi$ and $e$ are both transcendental, as are $\sin (\alpha), \cos (\alpha)$, and $\tan (\alpha)$, for any nonzero algebraic value of $\alpha$.*
- Likewise $\ln (a)$ and $a^{b}$ if $a$ is algebraic (and not 0 or 1 ), and $b$ is irrational algebraic.
* Note that $\alpha$ represents an angle in radians, not degrees.

Algebraic Numbers are Enumerable

- We define, for any algebraic equation $a_{n} x^{n}+a_{n-1} x^{n-1}+\ldots+a_{1} x+a_{0}=0$
the rank (or height) of the equation as:

$$
n+\left|a_{n}\right|+\left|a_{n-1}\right|+\ldots+\left|a_{1}\right|+\left|a_{0}\right|
$$

- Since all the coefficients and powers are integers, clearly there are a finite number of algebraic equations of any given rank $r$.
- These can be ordered, e.g. first by $n$, then by $a_{n}$, then by $a_{n-1}, \ldots$, finally by $a_{0}$.

E Each algebraic equation has at most $n$ solutions (where $n$ is the highest power of $x$ in the equation); and we can arrange these in numerical order.*

- So now we can make a list that will include all the algebraic numbers, ordered first by the equation to which they are a solution (as explained earlier), and then by the value of the solution.
- As before, we can define a surjection from $\mathbb{N}$ which maps 1 to the $1^{\text {st }}$ in the list, 2 to the $2^{\text {nd }}, 3$ to the $3^{\text {rd }}$, and so on. QED! (The Latin "quod erat demonstrandum" means "what was to be proved".)
* If complex numbers were to be included, we could order the solutions first by real part, then by imaginary part. 28

28

## Real Numbers are Not Enumerable

E Suppose that the set of real numbers $\mathbb{R}$ is enumerable. Then all $x \in \mathbb{R}$ (where $0<x<1$ ) can be put into an infinite list:
$a_{1}: \quad 0 . a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} \ldots$
$a_{2}: \quad 0 . a_{21} a_{22} a_{23} a_{24} a_{25} a_{26} a_{27} a_{28} \ldots$
$a_{3}: \quad 0 . a_{31} a_{32} a_{33} a_{34} a_{35} a_{36} a_{37} a_{38} \ldots$
$a_{4}: \quad 0 . a_{41} a_{42} a_{43} a_{44} a_{45} a_{46} a_{47} a_{48} \ldots$
$a_{5}: \quad 0 . a_{51} a_{52} a_{53} a_{54} a_{55} a_{56} a_{57} a_{58} \ldots$
30
30

## Turing on Computability and Intelligence

## Diagonalisation

- Consider now the number $c$ (for Cantor) that we get if we take the digits in order down the long diagonal of this list, adjusting them so that always $c_{n} \neq a_{n n}$, e.g. using this simple rule:

If $a_{n n}=5$, then $c_{n}=0$, otherwise $c_{n}=5$.

- Clearly the number $c$ will differ from the $n^{\text {th }}$ number in the list at the $n^{\text {th }}$ decimal place, for all $n$. Hence $c$ is a real number between 0 and 1 that does not occur in the list of all such numbers ... CONTRADICTION!!

Suppose $F$ is a surjection from $A$ to $\}(A)$, where $A$ is any enumerable set, then we list $A$ 's elements horizontally, and their corresponding subsets vertically, with ticks or crosses showing set membership:


Now consider the set of elements that have crosses down the long diagonal. Where is this set in the list? It should have a cross where there is a tick on the long diagonal, and a tick where there is a cross. So it differs in the $n^{\text {th }}$ place from the $n^{\text {th }}$ set in the list.

E Cantor's "diagonal" argument was the first example of a clever type of argument that works by taking two dimensions of variation and then providing some construction that runs "down the diagonal".

- Another famous example is the proof of Cantor's Theorem that for any set $A$, the power set $\wp(A)$ or $2^{A}$, i.e. the set of all subsets of $A$, always has a greater cardinality than $A$ itself.
- This is obvious for finite sets (where a set of $n$ objects has $2^{n}$ possible subsets), but Cantor proved that it is true for all sets, deriving a contradiction from the assumption of a surjective function from any set to its power set.


## An Infinity of Infinities

- The proof of Cantor's Theorem does not rely on enumerability of the domain of $F$, and can work without the explicit diagonalisation, if we simply consider the set:

$$
S=\{x \in A: x \notin F(x)\}
$$

- $S$ cannot be in the range of $F$, because if $\mathrm{S}=F(x)$ for any $x$, then $x \in \mathrm{~S} \leftrightarrow x \notin \mathrm{~S}$.
- This implies that the power set of any infinite set must have a cardinality strictly greater than that set - we cannot stop at $2{ }^{\aleph_{0}}$ (which is the cardinality of the continuum - Petzold pp. 31-2).


## Russell's Paradox

- By contemplating Cantor's argument, Russell came to his famous paradox of

The set of all sets that are not members of themselves

- This is similar to the "paradox" of the village barber who shaves all and only those men who do not shave themselves. But whereas the latter can easily be avoided (by denying that there is such a barber, or that the barber is a man), Russell's paradox demonstrates a contradiction in the heart of the logical theories that he and Frege had been attempting to develop.
- Russell to Frege, 16 June 1902:
"Let $w$ be the predicate of being a predicate which cannot be predicated of itself. Can $w$ be predicated of itself? From either answer follows its contradictory. ... Likewise, there is no class (as a whole) of those classes which, as wholes, are not members of themselves. From this I conclude that under certain circumstances a definable set does not form a whole."
- Frege to Russell, 22 June 1902:
"Your discovery of the contradiction has surprised me beyond words and, I should almost like to say, left me thunderstruck, because it has rocked the ground on which I meant to build arithmetic."


## Alan Turing on Computability and Intelligence

## Lecture 2

Hilbert's Programme and Gödel's Theorem

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy Hertford College, Oxford


37

## Consistency, Completeness, and Decidability

- Consistency
- The set of axioms should be provably consistent.
- Completeness:
- All mathematical truths should (in principle) be deducible from those axioms.
- Decidability:
- There should be a clearly formulated procedure which is such that, given any statement of mathematics, it can definitively establish within a finite time whether or not that statement follows from the given axioms.
39

39

## Consistency and Completeness

E Thus consistency (cannot prove both $P$ and $\neg P$ ) and completeness (can prove either $P$ or $\neg P$ ) are closely related, and can be understood quite independently of whether or not the axioms are true and the rules valid (i.e. truth-preserving).

- If, however, we were able to achieve a consistent and complete system of arithmetic, with true axioms and valid rules, then any arithmetical proposition would be provable if, and only if, it is true. A major part of Hilbert's dream would thus be realised.


## Hilbert's Programme <br> (from lecture 1, slide 12)

- Recall Hilbert's ambitions:
- 1921: To establish mathematics on a solid and provably consistent foundation of axioms, from which, in principle, all mathematical truths can be deduced (by the standard rules of first order predicate logic - this will usually be assumed in what follows).
- 1928: the Entscheidungsproblem or "decision problem": can an effective procedure be devised which would demonstrate - in a finite time - whether any given mathematical proposition is, or is not, provable from a given set of axioms?


38

## Syntactic Formalism

- In formal treatments, these notions are standardly interpreted syntactically (i.e. in terms of structural relationships between formulae) rather than semantically (i.e. in terms of truth and meaning).
E Thus understood,
- a consistent system is one in which it is never possible to prove both a proposition P and its negation $\neg$ P;
- a complete system is one in which it is always possible either to prove P or to prove $\neg \mathrm{P}$, for any proposition P that is expressible within the system.


## Introducing Recursive Functions

- Many functions can be defined by specifying the result for higher-value arguments in terms of results for lower-value arguments. Here we define factorial (in terms of addition and multiplication):

$$
\begin{aligned}
& 0!=1 \\
& (x+1)!=x!.(x+1)
\end{aligned}
$$

- The same can be done for two-place or multi-place functions, e.g. multiplication (in terms of addition):
$x .0=0$
$x \cdot(y+1)=(x \cdot y)+x$
42
42


## Turing on Computability and Intelligence

- We can dig down even further, defining addition in terms of the successor relation (" $S$ "):

$$
\begin{aligned}
& x+0=x \\
& x+S(y)=S(x+y)
\end{aligned}
$$

- A natural question to ask is: what class of functions is definable in this sort of way, starting from a minimal base of primitive operations:
- The zero function $Z(x)=0$
- The successor function $S$
- Projection or identity functions (e.g. the function $P_{i}^{n}$ which takes $n$ arguments and returns the value of the $i^{\text {th }}$ of them, e.g. $P_{2}^{4}(12,34,56,67)=34$
- Function composition (i.e. applying functions in turn)


## Gödel's Theorem (1931)

- Gödel's First Incompleteness Theorem: In any true (and hence consistent) axiomatic theory sufficiently rich to enable the expression and proof of basic arithmetic propositions (call this "PM"),
- It will be possible to construct an arithmetical proposition $G$ such that neither $G$, nor its negation, is provable from the given axioms.
- Therefore the system must be incomplete.
- Moreover it follows from Gödel's reasoning - on the assumption that the system is indeed true (and hence consistent) - that G must, in fact, be a true statement of arithmetic. Can we then know unprovable truths?


## The Proof Strategy

1. Devise a systematic method for assigning a "Gödel number" $g[F]$ to every formula $F$ - and every sequence of formulae - that are expressible within PM.
2. Express logical relationships (e.g. "sequence $S$ is a proof of formula $F^{\prime \prime}$ ) in terms of mathematical relationships between the Gödel numbers of $S$ and $F$ : the mathematical formula expressing that relationship, $\operatorname{Proof}(g[S], g[F])$ for short, will be true if, and only if, $S$ is a valid proof of $F$.
3. Devise a mathematical formula $G$ which, according to this method, is true if, and only if, there is no sequence $S$ which yields a valid proof of $G$ itself. This formula will be of the form " $\neg \exists x \operatorname{Proof}(x, g[G])$ ", i.e. "There is no $x$ such that $x$ is the Gödel number of a sequence that proves $G^{\prime \prime}$...

## Gödel Numbering

- Gödel's proof encodes statements about mathematical relationships (e.g. that some sequence of formulae provides a valid proof of some formula $F$ ) as formulae within arithmetic.
- This involves assigning a Gödel number, g[f] for short, to each formula $f$, according to its structure (following Nagel and Newman, revised by Hofstadter, NYU 2001): - Constant symbols have $\mathbf{g}$ numbers 1 ( " $\neg$ "), 2 (" v "), 3 (" $\rightarrow$ "), 4 (" $\mathrm{J}^{\text {"), }}$ 5 ("="), 6 ("0"), 7 ("s"), 8 ("("), 9 (")"), 10 (","), 11 ("+"), and 12 (" $\times$ "). ■ Numerical variables have $\boldsymbol{g}$ numbers 13 (" $x$ "), 17 ( " $y$ "), $19,23, \ldots$ - Sentential variables have $\boldsymbol{g}$ numbers $13^{2}$ (" $p$ "), $17^{2}$ (" $q$ "), $19^{2}, \ldots$ - Predicate variables have $\mathbf{g}$ numbers $13^{3}$ ("P"), $17^{3}$ (" $Q$ "), $19^{3}$, .. (13, 17, 19, $23 \ldots$ is following the sequence of prime numbers >12)


## Turing on Computability and Intelligence

Having Gödel numbers for the individual symbols, a number can now be worked out for each formula by raising successive prime numbers to those powers:
Then a Gödel number for a sequence of formulae can be derived similarly, by raising 2 to the power of the first formula, 3 to the power of the second, and so on, then multiplying these all together. Obviously, the numbers are astronomical!

* Note that "s" is the sign for "successor", e.g. 1=s0.


## Decoding Gödel Numbers

E Note that each formula (or sequence of formulae) has a unique Gödel number - no two formulae (or sequences) can have the same Gödel number.
E Moreover it is straightforward "in principle" to decode a Gödel number, by factorising it into its prime factors, examining the powers of those factors, etc.

- This makes it feasible to use Gödel numbers as proxies for those formulae in expressing their properties and the relations between them (e.g. the function we shall call sub that holds when one formula is a substitution instance of another).

50

## Abbreviated Expressions

For ease of exposition, we allow ourselves to use abbreviations, so for example:
" $p \wedge q$ " is understood as short for " $\neg(\neg p \vee \neg q)$ "
" 1 " is understood as short for " s 0 "
" 2 " is understood as short for "ss0"
" 3 " is understood as short for "sss0" (etc.)

- The Gödel number of a formula containing one or more abbreviations is defined to be the Gödel number of the expanded formula after replacement of the abbreviations.

52

- So here we have taken formula number:

$$
\begin{gathered}
2^{4} \times 3^{13} \times 5^{8} \times 7^{13} \times 11^{5} \times 13^{7} \times 17^{17} \times 19^{9} \\
\text { " } \exists x(x=\mathrm{s} y) "
\end{gathered}
$$

and within it we have substituted the variable with number:

$$
\text { " } y "
$$

by the numerical expression for the number: 2 "ss0"

E This substitution has yielded formula number:

$$
2^{4} \times 3^{13} \times 5^{8} \times 7^{13} \times 11^{5} \times 13^{7} \times 17^{7} \times 19^{7} \times 23^{6} \times 29^{9}
$$

$$
" \exists x(x=\operatorname{sss} 0) "
$$

54
54

## Turing on Computability and Intelligence

E There will be a particular arithmetical function relating these four numbers. We can write:

```
24}\times\mp@subsup{3}{}{13}\times\mp@subsup{5}{}{8}\times\mp@subsup{7}{}{13}\times1\mp@subsup{1}{}{5}\times1\mp@subsup{3}{}{7}\times1\mp@subsup{7}{}{7}\times1\mp@subsup{9}{}{7}\times2\mp@subsup{3}{}{6}\times2\mp@subsup{9}{}{9
    = sub (24}\times\mp@subsup{3}{}{13}\times\mp@subsup{5}{}{8}\times\mp@subsup{7}{}{13}\times1\mp@subsup{1}{}{5}\times1\mp@subsup{3}{}{7}\times1\mp@subsup{7}{}{17}\times1\mp@subsup{9}{}{9},17,2
```

- So sub is an arithmetical function with 3 inputs:
- the Gödel number of a formula containing a variable;
- the Gödel number of that variable (e.g. 17 for " $y$ ");
- a number (e.g. 2) whose numeral (e.g. "ss0") is to be substituted for that variable

Given these inputs, sub yields:

- the Gödel number of the formula resulting from substitution of the variable with the numeral.

Sub is a complicated function of PM, but recursively well-defined and could be spelled out in detail.

- If $F_{y}$ is a formula containing $y$ (Gödel number 17) and $F_{n}$ is the corresponding formula in which $y$ is substituted by the numeral for $n$, then we have:
$\mathrm{g}\left[F_{n}\right]=\operatorname{sub}\left(\mathrm{g}\left[F_{y}\right], 17, n\right)$
- Imagine now that the function $\operatorname{Sub}(a, b, c)$ is spelled out in gory arithmetical detail in PM, and that we write out this expression with $y$ in place of $a$ and $c$, and 17 in place of $b$. This yields a complicated (but provably possible) arithmetical expression for: $\operatorname{Sub}(y, 17, y)$ which will soon play a key role.


## Arithmetising Meta-Mathematics

E sub is a relatively simple function. But Gödel also showed that it is possible to define a (far more complicated) arithmetical formula $A$ corresponding to the meta-mathematical statement that a sequence of formulae $S$ constitutes a proof of formula $F$. A will be (arithmetically) true if and only if $S$ indeed proves $F$.

- Let us use " $\operatorname{Proof}(a, b)$ " as shorthand for the arithmetical formula in PM corresponding to:
"The sequence of formulae with Gödel number $a$ is a proof (or demonstration)* of the formula with Gödel number $b$."
*Nagel and Newman use "Dem" instead of "Proof"

57

## The Heart of Gödel's Argument

- Consider now the arithmetical formula in PM: $\neg \operatorname{Proof}(a, b)$
- This corresponds to the statement that the sequence of formulae with Gödel number $a$ is not a proof of the formula with Gödel number $b$.

E Likewise this arithmetical formula: *
$\neg \exists x \operatorname{Proof}(x, \operatorname{Sub}(y, 17, y))$
Corresponds to the statement that there is no proof of the formula with Gödel number $\operatorname{sub}(y, 17, y)$.

* Nagel \& Newman originally used " $(x) \neg$ " instead of " $\neg \exists x$ "; we here follow Hofstadter's 2001 revised edition (as in the numeric coding)


## A Truth-Preserving Correspondence

E The crucial point here is that the arithmetical formula $\operatorname{Proof}(a, b)$ in PM will be true if, and only if, the metamathematical statement "The sequence of formulae with Gödel number $a$ is a proof of the formula with Gödel number $b$ " is also true. [Note that Proof $(a, b)$ expresses a statement - e.g. of an arithmetical equation - rather than a mere numerical expression like $\operatorname{Sub}(a, b, c)]$

- So to establish whether or not the sequence of formulae with Gödel number $a$ is in fact a valid proof of the formula with Gödel number $c$, it suffices to establish whether or not the numbers $a$ and $c$ yield a true equation when substituted to give the arithmetical formula $\operatorname{Proof}(a, c)$.

58

$$
\neg \exists x \operatorname{Proof}(x, \operatorname{Sub}(y, 17, y))
$$

- This arithmetical formula corresponds to the meta-mathematical statement that
$-\ldots$ there exists no sequence of formulae that constitutes a proof of the formula with Gödel number $\operatorname{sub}(y, 17, y)$.
- ... or in other words, that formula (whose precise identity will obviously depend on the value substituted for " $y$ ") is not provable.
- Now consider the arithmetical formula above, and suppose it has the Gödel number $n$.


## Gödel’s Magical Move

- We have that:

9[ $\neg \exists x \operatorname{Proof}(x, \operatorname{Sub}(y, 17, y))]=n$

- Now consider the formula:
(G) $\quad \neg \exists x \operatorname{Proof}(x, \operatorname{Sub}(n, 17, n))$
and notice that this is itself the formula that we obtain from the formula with Gödel number $n$ if we substitute $y$ by the numeral for $n$. Hence:
$9[\neg \exists x \operatorname{Proof}(x, \operatorname{Sub}(n, 17, n))]=\operatorname{sub}(n, 17, n)$
61
(G) $\neg \exists x \operatorname{Proof}(x, \operatorname{Sub}(n, 17, n))$
- This arithmetical formula corresponds to the meta-mathematical statement that
- ...there exists no sequence of formulae that constitutes a proof of the formula with Gödel number $\operatorname{sub}(n, 17, n)$.
$-\ldots$ or in other words, that formula is not provable.
- But this arithmetical formula itself has the Gödel number $\operatorname{sub}(n, 17, n)$ !
- So $G$ corresponds to the meta-mathematical statement that $G$ itself is unprovable!!

62

To spell this out further, the encoding means that:

$$
G \text { is true } \leftrightarrow G \text { is unprovable }
$$

$\therefore \quad \neg G \rightarrow G$ is provable
But if $G$ is provable and the system provides a faithful and consistent representation of arithmetic, then $G$ must be true, so we have:

$$
\neg G \rightarrow G \text { is provable } \rightarrow G
$$

So $G$ cannot be false, and hence must be true. Yet if it is true, it is unprovable (because it encodes the statement that it is unprovable). So our system, if it is a consistent and correct axiomatisation of arithmetic, cannot be complete, for $G$ will then be a true statement of arithmetic that cannot be proven. 64

64

## What is "effective computability"?

- An "effectively computable" procedure is supposed to be one that:
- can be performed by systematic application of clearly specified rules,
- without requiring any inspirational leaps or spontaneous intellectual insights.

E So to find the limits of "effective computability",

- we need to devise a way of encompassing all possible mechanical methods of inference ...
- ... and this is how Alan Turing came to invent what is now known as the Turing machine.


## Alan Turing on Computability and Intelligence

## Lecture 3

"On Computable Numbers": Turing's 1936 Paper

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy

Hertford College, Oxford


## Effective Computability

- The key contribution of Alan Turing's 1936 paper
"On Computable Numbers, with an application to the Entscheidungsproblem"
(Proceedings of the London Mathematical
Society, $2^{\text {nd }}$ series, Vol. 42, pp. 230-65)
is to provide a characterisation of "effective computability" in terms of the behaviour of an extremely simple type of machine which, he argues, can execute (either directly or indirectly) all possible methods of "mechanical" information processing This is now universally known as the Turing Machine.

68
§ 1 Computing Machines (p. 68)

- Introducing the Turing machine
- Most of (the short) § 1 is concerned with explaining the structure of a Turing machine. But first ...

E Justification of the Turing machine

- "We have said that the computable numbers are those whose decimals [binimals] are calculable by finite means. ... the justification lies in the fact that the human memory is necessarily limited."
- Turing refers forward to $\S 9$, where he will argue in outline why any finitely calculable number should be calculable by a Turing machine ...

70

Note here that the "computer", for Turing, is the person performing the computation:
"The behaviour of the computer at any moment is determined by the symbols which he is observing, and his 'state of mind' at that moment. We may suppose that there is a bound $B$ to the number of symbols or squares which the computer can observe at any one moment. ... We will also suppose that the number of states of mind which need be taken into account is finite. ... If we admitted an infinity of states of mind, some of them will be 'arbitrarily close' and will be confused. ..." (Petzold, p. 191)

## Turing on Computability and Intelligence

The computer's repertoire of operations is precisely circumscribed in terms of "simple operations" that cannot be further divided:"Let us imagine the operations performed by the computer to be split up into 'simple operations' ... Every such operation consists of some change of the physical system consisting of the computer and his tape. ... We may suppose that in a simple operation not more than one symbol is altered. ... We may ... without loss of generality, assume that the squares whose symbols are changed are always 'observed' squares." (Petzold, p. 192)
"The most general single operation must therefore be taken to be one of the following:
(A) A possible change of symbol together with a possible change of state of mind.
(B) A possible change of observed squares, together with a possible change of state of mind.

The operation actually performed is determined, as has been suggested [highlighted passage in slide 72], by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after
"Besides these changes of symbols, the simple operations must include changes of distribution of observed squares. The new observed squares must be immediately recognisable by the computer. ... it is reasonable to suppose that they can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount ... say ... $L$ squares ...

It may be that some of these changes [i.e. of symbol or of observed squares] necessarily involve a change of state of mind."
(Petzold, p. 192-3)
"We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an ' $m$-configuration' of the machine. The machine scans $B$ squares corresponding to the $B$ squares observed by the computer. In any move the machine can change a symbol on a scanned square or can change any one of the scanned squares to another square distant not more than $L$ squares from one of the other scanned squares. The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration."
(Petzold, p. 194)

## The Turing Machine comprises:

- A potentially infinite tape divided into numbered squares, on which symbols can be written/erased.
- A read/write head (as in a tape recorder) which can "scan" symbols and also write (or erase) them.
- An "m-configuration" (usually now called a "state"), taking any of a specified finite range of values this is the machine's active memory. Computation starts from the first, initial, state.
- Instructions in the form of a "machine table", which tell it what to do in each possible circumstance ...


## Turing on Computability and Intelligence

## A Turing Machine in Action

Here we see a Turtle System Turing Machine simulator, at 45 cycles through the irrational (probably transcendental) number program that Turing gives in § 3 of his paper (Petzold, p. 87).
79


79

## The Repertoire of Actions (pp. 70-1)

- At each stage, depending on the configuration (i.e. the symbol read from the current square, and the current state), the machine can:
- Erase the symbol, or write a new one;
- Move the scanning head one place left or right on the tape;
- Change to a new state ( $m$-configuration).
- Initially, Turing allows multiple actions to be treated as one (though this is restricted when the machines come to be regimented at §5).

81

E/F Squares, and Marking (pp. 93-4)
In § 3 Turing explains that his machines will write figures only on alternate squares:

- These are called " $F$-squares" (figure);
- The others are "E-squares" (erasable), and will be used for working notes.
- "If a symbol $\beta$ is on an $F$-square $S$ and a symbol $\alpha$ is on the $E$-square next on the right of $S$, then $S$ and $\beta$ will be said to be marked with $\alpha$. The process of printing this $\alpha$ will be called marking $\beta$ (or $S$ ) with $\alpha^{\prime \prime}$ (end of § 3 ).


## The "Configuration" (pp. 69-70)

- At any point, the machine can scan only one square on the tape (the "current" or "scanned" square).
E As it moves left and right on the tape (one square at a time), it scans the symbol on the new current square, and takes account of this in its behaviour.
- Turing refers to the combination of
- the $m$-configuration (what we call "state")
- the "scanned symbol"
as the current "configuration" of the machine. The next action of the machine depends entirely on this.


## § 2 Definitions (p. 72)

E Turing focuses on automatic machines, whose behaviour is entirely determined by the configuration (though he also mentions choice machines, which are not fully determined).
E Any machine will have a limited range of symbols that it can read and write. Turing distinguishes:

- figures: "0" and "1"
- other symbols ("symbols of the second kind")

82

## The Computed Number (p. 73)

- If an automatic machine starts from a blank tape in its initial state, then the sequence of figures (i.e. " 0 " and " 1 ") that it writes on the tape is called the "sequence computed by the machine".
E"The real number whose expression as a binary decimal (sic.) is obtained by prefacing this sequence by a decimal [binary] point is called the number computed by the machine."
- In Turing's machines, the binary digits of this number are supposed to be printed left-to-right on successive $F$-squares, and never erased.


## Turing on Computability and Intelligence

85

87

89

## An Example of a Marked Tape (p. 89)

- The tape is like this at one stage in the operation of the machine Turing specifies on p . 87:



## The Complete Configuration (p. 75)

E We have already seen " $m$-configuration" (i.e. state) and "configuration". But Turing defines yet another use of the word, intended to capture the complete state of a computation at any time:
"At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the m-configuration will be said to describe the complete configuration at that stage. The changes of the machine and tape between successive complete configurations will be called the moves of the machine."

87

## § 3 Examples of Computing Machines

- Turing first gives an illustration of a machine
table for a machine that will compute the binimal for one third:

0101010101 ...
In the full form (Petzold p. 81), this requires 4
states (starting from "b" for "begin"):
b ("b") Prints a " 0 " and moves right into state c
c ("c") Moves right into state e
e ("e") Prints a "1" and moves right into state $k$
$k$ (" $k$ ") Moves right into state b

## Simplifying the Tables

Turing points out that machine tables can be greatly simplified if we allow a single transition to perform multiple operations (p. 84):

| m-config. |
| :--- |
| bymbol |
| None P0 b <br> 0 $\mathrm{R}, \mathrm{R}, \mathrm{P} 1$ b <br> 1 $\mathrm{R}, \mathrm{R}, \mathrm{P} 0$ b <br> 0   |

## Circular and Circle-Free Machines (p. 76)

E A Turing machine that carries on writing out binary digits for ever in this way is called "circle-free": these define computable numbers.

- We are interested in binary fractions, expressed as non-terminating binimals (so sequences that correspond to rational numbers recur for ever).
- Later, Turing will call a number that defines such a circle-free machine a "satisfactory" number ( § 8).
- A Turing machine that stops writing out binary digits at any stage is called "circular". [Think of it as getting stuck in a non-writing loop or "circle".]
86
86


| m-config. | symbol | operations | final m-config. |
| :---: | :--- | :--- | :---: |
| b | None | P0, R | c |
| c | None | R | e |
| e | None | P1, R | k |
| k | None | R | b |

39


90

## Turing on Computability and Intelligence

## A More Complex Example

- Turing then gives a far more complex example, which computes the sequence: 0010110111011110111110 ... (see Petzold pp. 85-90 for details)
- This is irrational, probably transcendental.
- Note Turing's convention of marking the lefthand end of the tape (i.e. printing on the first $F$ and $E$ - squares) two "schwa" characters: "ӘӘ".
- Note also that two successive blanks always signify the right of the printed part of the tape.


## Listing Complete Configurations

- Turing illustrates the working of this machine with a sequence of "complete configurations" that include the entire non-blank portion of the tape, with the state and position indicated.
- The successive complete configurations are separated by colons, and Turing then puts the state name into the sequence also, just to the left of the scanned symbol, e.g. (Petzold p. 92):

$$
\text { b: Ә Ә о } 0 \quad 0 \text { : Ә Әq } 0 \text { 0: ... }
$$

- This format is flagged as "(C)", cf. Petzold p. 144.


## Getting Used to Turing Machines

- To familiarise yourself with how Turing machines work, see Chapter 6 of Petzold ("Addition and Multiplication").
E This departs from Turing's paper, giving two helpful examples:
- A machine to generate all the positive integers in sequence (pp. 99-100);
- A machine to calculate the square root of 2 (pp. 100-108).


## § 4 Abbreviated Tables (pp. 113-5)

- Initially, one of the most difficult aspects of Turing's paper is understanding his use of "skeleton tables" to define machines.

E These make use of " $m$-functions" which enable many different " $m$-configurations" (i.e. states) to be defined that have very similar behaviour but with slight differences, for example:

- handling different symbols (either to read/find or write/mark);
- moving to different states once their work is done.

Turing's First $m$-function "Find" (p. 115)

| m-config. | Symbol | Behaviour | Final m-config. |
| :---: | :---: | :---: | :---: |
| $\Gamma$ | $\vartheta$ | L | $f_{1}(C, B, a)$ |
| $f(C, B, \alpha)\{$ | not $Ө$ | L | $f(C, B, \alpha)$ |
| L | None * | L | $f(C, B, \alpha)$ |
| - | $\alpha$ |  | C |
| $\mathrm{f}_{1}(\mathrm{C}, \mathrm{B}, \mathrm{\alpha})$ | not $\alpha$ | R | $\mathrm{f}_{1}(\mathrm{C}, \mathrm{B}, \mathrm{a})$ |
| L | None | R | $\mathrm{f}_{2}(\mathrm{C}, \mathrm{B}, \mathrm{a})$ |
| - | a |  | C |
| $\mathrm{f}_{2}(\mathrm{C}, \mathrm{B}, \alpha)\{$ | not $\alpha$ | R | $\mathrm{f}_{1}(\mathrm{C}, \mathrm{B}, \mathrm{\alpha})$ |
| L | None | R | B |

[^0] is entered at that point; if not, state B is entered.


96

## Turing on Computability and Intelligence

## Alan Turing on Computability and Intelligence

## Lecture 4

Enumerating the Computable Numbers, and the Universal Turing Machine

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy
Hertford College, Oxford


97
98

## Summarising "Find"

- Since any machine that includes the $m$-function $f(C, B, \alpha)$ is guaranteed to reach state $C$ or $B$ when it exits from this nexus, we can simplify the diagram:

"Erase" (3 arguments, erase once)

| m-config. | Behaviour | Final m-config. |
| :---: | :---: | :---: |
| $e(C, B, \alpha)$ |  | $f\left(e_{1}(C, B, \alpha), B, \alpha\right)$ |
| $\mathrm{e}_{1}(C, B, \alpha)$ | $E$ | $C$ |

Turing (p. 118) defines two versions of this $m$-function, first with three arguments (shown above), then with two. On going into state $e(C, B, \alpha)$, the machine transitions into state $f\left(\mathrm{e}_{1}(C, B, \alpha), B, \alpha\right)$, which searches for the leftmost " $\alpha$ " and enters state $\mathrm{e}_{1}(\mathrm{C}, \mathrm{B}, \mathrm{\alpha})$ on the square containing that " $\alpha$ " (or ends in state B if there is no " $\alpha$ "). Then it erases that square (action E) and enters state C.

100

99
100
"Erase" (2 arguments, repeated erase)


Note how State e(B, a) immediately transitions to state $e(e(B, \alpha), B, \alpha)$, which is defined so that after erasing the leftmost " $\alpha$ ", it reverts to state $e(B, \alpha)$, and repeats.

## Turing on Computability and Intelligence

## The "Print at the End" m-function



103

The "Copy" $m$-function


105

## A Summary of "Subroutines"

- The main tape-manipulating "subroutines" defined in this way in § 4 are:
f find first " $\alpha$ " ( $f$ '/f" then move left/right) e erase first " $\alpha$ " (or all " $\alpha$ "s, or all markers) pe print " $\beta$ " in the first blank $F$-square C copy into the first blank $F$-square the first symbol marked with an " a "
ce copy at the end all the symbols marked with an " $\alpha$ ", then erase the " $\alpha$ " $s$
re replace first " $\alpha$ " (or all " $\alpha$ " $s$ ) with " $\beta$ "
"Left", "Right", f' and f" $m$-functions

| m-config. | Behaviour | Final m-config. |
| :---: | :---: | :---: |
| I(C) | L | $C$ |
| $\mathrm{r}(\mathrm{C})$ | R | C |

The table has only three columns, as this behaviour is unaffected by tape symbols (p. 121)


#### Abstract

$f^{\prime}(C, B, a)$ $f(1(C), B, a)$ $f(r(C), B, \alpha)$ finds the left$\mathrm{f}^{\prime \prime}(\mathrm{C}, \mathrm{B}, \mathrm{a})$ f(r(C), B, a) most " $a$ " then enters state $r(C)$, thus moving right and entering state C. So $f^{\prime \prime}(C, B, \alpha)$ is similar in effect to $f(C, B, \alpha)$, except that it moves right after finding the leftmost " $\alpha$ ". 104


104

## The "Compare" m-function

| m-config. | Symbol | Behaviour | Final m-config. |
| :---: | :---: | :---: | :---: |
| $\operatorname{cp}(\mathrm{C}, \mathrm{A}, \mathrm{E}, \alpha, \beta)$ |  |  | $\mathrm{f}^{\prime}\left(\mathrm{cp}_{1}(\mathrm{C}, \mathrm{A}, \beta), \mathrm{f}(\mathrm{A}, \mathrm{E}, \beta), \alpha\right)$ |
| $\mathrm{cp}_{1}(\mathrm{C}, \mathrm{A}, \beta$ ) | Y |  | $\mathrm{f}^{\prime}\left(\mathrm{cp}_{2}(\mathrm{C}, \mathrm{A}, \mathrm{Y}), \mathrm{A}, \beta\right)$ |
| $\mathrm{cp}_{2}(\mathrm{C}, \mathrm{A}, \mathrm{Y})$ | Y |  | C |
| $\mathrm{cp}_{2}(\mathrm{C}, \mathrm{A}, \mathrm{Y})$ | not Y |  | A |

Turing ( p .123 ) explains the behaviour of this $m$-function thus: "The first symbol marked $\alpha$ and the first marked $\beta$ are compared. If there is neither $\alpha$ nor $\beta, \rightarrow E$.* If there are both and the symbols are alike $\rightarrow \mathrm{C}$. Otherwise $\rightarrow \mathrm{A}$."

* note that the Gothic " E " looks very like the "C"! 106

106
cr copy at the end all the symbols marked with an " $\alpha$ " (without erasing the " $\alpha$ "s)
$\mathrm{cp} \quad$ compares the first symbol marked " $\alpha$ " with the first symbol marked " $\beta$ ", entering different next states depending on the outcome
cpe likewise, but erase markers if similar (or compare two sequences ...)
g find last " $\alpha$ " (the paper calls this "q", but " $g$ " seems intended - see p. 124)
$\mathrm{pe}_{2} \quad$ print 2 characters in two $F$-squares
$\mathrm{ce}_{2}, \mathrm{ce}_{3}$ copy to the end symbols marked $\alpha, \beta$ (and Y ), erasing the symbols

## Turing on Computability and Intelligence

## § 5 Enumeration of Computable Sequences

E Turing now explains (pp. 131 ff.) how to put all possible machine tables into a standard form, ultimately reducing each to a single number.

- We convert each table into a table that has one write and one move/nomove (L/R/N) per line;
- We number all the states ( $q_{1}, q_{2}$ etc.), and all the symbols $\left(S_{0}, S_{1}, S_{2}\right.$ etc. $-S_{0}$ is the blank);
- Each line of the table now takes one of the forms " $q_{i} S_{j} S_{k} L q_{m}$ ", " $q_{i} S_{j} S_{k} R q_{m}$ ", or " $q_{i} S_{j} S_{k} N q_{m}$ ".
- These can be listed, separated by semicolons. 109

109

Finding the Description Number of a Machine: Turing's Example (pp. 138-40)
"Let us find a description number for the machine I of § 3 ."

This is the machine (p. 81, slide 83) that prints out the binimal "0 10101 ...":


111

Other tables could be obtained by adding irrelevant lines such as [the last one here]:"

| $q_{1}$ | $S_{0}$ | $\mathrm{P} S_{1}, \mathrm{R}$ | $q_{2}$ |
| :---: | :---: | :---: | :---: |
| $q_{2}$ | $S_{0}$ | $\mathrm{P} S_{0}, \mathrm{R}$ | $q_{3}$ |
| $q_{3}$ | $S_{0}$ | $\mathrm{P} S_{2}, \mathrm{R}$ | $q_{4}$ |
| $q_{4}$ | $S_{0}$ | $\mathrm{P} S_{0}, \mathrm{R}$ | $q_{1}$ |
| $q_{1}$ | $S_{1}$ | $\mathrm{P} S_{1}, \mathrm{R}$ | $q_{2}$ |

This final line never comes into play, as the machine always moves right, so the current symbol is always blank ( $S_{0}$ ).

| m-config. | symbol | operations | final m-config. |
| :---: | :--- | :--- | :---: |
| b | None | PO, R | c |
| c | None | R | e |
| e | None | P1, R | k |
| k | None | R | b |

"When we rename the $m$-configurations its table becomes:
$\left.\begin{array}{|c|c|c|c|}\hline & q_{1} & S_{0} & \mathrm{P} S_{1}, \mathrm{R} \\ & q_{2} & S_{0} & \mathrm{PS}_{0}, \mathrm{R}\end{array}\right) q_{3}$.

112

- Turing has illustrated in passing here that many different machines can be entirely equivalent in behaviour. Adding an "irrelevant" line to the table (i.e. one that is never actually invoked) will make no difference at all to the behaviour, and hence will generate the same sequence.
- It is also possible to have other machines that generate the same sequence by means of different state transitions: that are equivalent in output, but not trivially so.
- Thus the mapping from "satisfactory" machines to computable numbers is surjective (each such machine generates just one computable number) but it is not injective.

Having explained his two illustrative tables, Turing continues (p. 139):<br>"Our first standard form would be<br>$q_{1} S_{0} S_{1} R q_{2} ; q_{2} S_{0} S_{0} R q_{3} ; q_{3} S_{0} S_{2} R q_{4} ;$<br>$q_{4} S_{0} S_{0} R q_{1}$; .<br>The standard description is<br>DADDCRDAA;DAADDRDAAA; DAAADDCCRDAAAA;DAAAADDRDA;"<br>\[ \begin{aligned} \& {\left[" q_{1} " \Rightarrow " D A ", " S_{0} " \Rightarrow " D ", " S_{1} " \Rightarrow " D C ",\right.}<br>\& " q_{2} " \Rightarrow " D A A ", " q_{3} " \Rightarrow " D A A A ", etc.] \end{aligned} \]

## The Computable Numbers Are Enumerable

E "To each computable sequence there corresponds at least one description number, while to no description number does there correspond more than one computable sequence. The computable sequences and numbers are therefore enumerable." (p. 138)

- This is a significant result: it follows that "nearly all" of the real numbers in Cantor's universe are not computable.


## § 6 The Universal Machine

E In § 6 of his paper (pp. 143-9), Turing explains how to design "a single machine which can be used to compute any computable sequence.

If this machine $\mathscr{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathscr{M}_{\text {, }}$, then $\mathscr{U}$ will compute the same sequence as

E As Petzold remarks, Turing starts rather oddly:
"Let us first suppose that we have a machine $\mathscr{M}^{\prime}$ which will write down on the $F$-squares the successive complete configurations of $\mathscr{M}$."

## Generating Complete Configurations

- Recall from slide 92 how Turing lists sequences of "complete configurations" that include the entire non-blank portion of the tape, with the state and position indicated, e.g. (p. 92):

$$
\begin{equation*}
\text { b: Ә Ә о } 0 \quad 0 \text { : Ө Ә q } 0 \text { 0: ... } \tag{C}
\end{equation*}
$$

- At pp. 144-6, Turing returns to and modifies this format by replacing the state codes ( $\mathrm{b} \circ \mathrm{q}$ ) with "DA", DAA", and "DAAA", blanks by "D", "0" by "DC", "1" by "DCC", and " $Ә$ " by "DCCC", to fit with the standard description.


## Turing on Computability and Intelligence

Machine $\mathscr{M}^{\prime}$<br>- The result of these substitutions into (C) is:<br>"DA: DCCCDCCCDAADCDDC:<br>DCCCDCCCDAAADCDDC : ... ( $\mathrm{C}_{1}$ )<br>(This is the sequence of symbols on $F$-squares.)"<br>- Turing (p. 146) means that the machine $\mathscr{M}^{\prime}$, which is being designed to print out the successive configurations of machine $\mathscr{M}$, is to do so in this form (and on the F-squares).<br>- He remarks that "if $\mathscr{M}$ can be constructed, then so can $\mathscr{M}^{\prime}$." It would operate by referring back to a copy of the S.D. of $\mathscr{M}$, written on the tape.

## Identifying the Configuration

Here the configurations in $\left(\mathrm{C}_{1}\right)$, as generated by [non-standard] Machine II (p. 87), are underlined:
"DA : DCCCDCCCDAADCDDC :
DCCCDCCCDAAADCDDC : ..."
(this sequence is built up at the right of the tape)

- Recall that the complete configurations are separated by colons, and within them, just one state (represented by "D" followed by a sequence of "A"s) will appear, followed by the "scanned symbol" on the current square (represented by "D" followed by a sequence of "C"s).
- Here and in the next couple of slides, you might find it helpful to refer to Turing's text on p. 151.


## Identifying the Matching Rule

E We can underline the "trigger" configurations within the standard description of Machine I (pp. 81, 139-40) (note we can't do this for Machine II, as it's non-standard):
"DADDCRDAA;DAADDRDAAA; DAAADDCCRDAAAA; DAAAADDRDA"
(this is put at the left of the tape to start with)

- Overall, the first quintuple has been translated thus:

| Initial state | Read symbol | Write symbol | Move | Final state |
| :---: | :---: | :---: | :---: | :---: |
| b | None | 0 | $R$ | c |
| $q_{1}$ | $S_{0}$ | $S_{1}$ | $R$ | $q_{2}$ |
| $D A$ | $D$ | $D C$ | $R$ | $D A A$ |

124

DA : DCCCDCCCDAADCDDC: DCCCDCCCDAAADCDDC : ... ( $\left.\mathrm{C}_{1}\right)$ becomes:

DA: 0: 0: DCCCDCCCDAADCDDC : DCCCDCCCDAAADCDDC : ... ( $\mathrm{C}_{2}$

- So now, machine $\mathscr{M}^{\prime}$ will print out exactly the same computable number as machine $\mathscr{M}$, except that whereas $\mathscr{M}$ prints the figures [digits] on successive F -squares, most of the F -squares in the output of $\mathscr{I}^{\prime}$ will contain " $D$ ", " $C$ ", " $A$ ", or ":", with the occasional " 0 " or " 1 " interspersed.


#### Abstract

Achieving Universality E Machine $\mathscr{M}^{\prime}$ has been carefully designed to generate exactly the same sequence of figures as machine $\mathscr{M}$, by reading the relevant instructions from the standard description of machine $\mathscr{M}$ (as provided, from the start, at the left of the tape). Hence replacing the standard description of $\mathscr{M}$ (at the left of the tape) with the standard description of a different machine $\mathscr{N}$ will mean that we end up with the sequence of figures that $\mathscr{N}$ would generate on the tape, instead of the sequence of figures that $\mathscr{M}$ would generate


The set of states (m-configurations) available to the universal machine must include all those explicitly mentioned in the relevant tables, "together with all those which occur when we write out the unabbreviated tables" of the relevant $m$-functions (p. 149).

- For example the $m$-function "e(anf)" is used, and this requires also a state " $\mathrm{e}_{1}(\mathrm{anf})$ " (p. 150).
E "When $\mathscr{2}$ is ready to start work the tape running through it bears ... the symbol $\partial$ on an $F$-square and again $\partial$ on the next $E$-square; after this, on $F$-squares only, comes the S.D of the machine [to be mimicked] followed by a double colon "::" (a single symbol, on an $F$-square). The S.D consists of a number of instructions, separated by semi-colons." (p. 150)

129 - Note Turing's use of the word "instructions" here (cf. p. 82).

## § 7 Detailed Description of the Universal Machine

- § 7 describes his Universal Machine in detail, making use of many of the "subroutine" functions whose skeleton tables were in $\S 4$.
- This was the first proof that there could be a "universal" programmable machine, capable of computing any number that we know how to compute, when given the recipe.
- By extension, it seems clear that any other computable function will be achievable.
- On page 151, Turing explains exactly how each "instruction" (i.e. represented line of the machine table) is constituted. This is by now familiar.
- He goes on to list the various symbols that the Universal Machine $\mathscr{U}$ is to be capable of printing on the tape. In addition to " $D$ ", " $C$ ", " $A$ ", ":", " 0 ", and " 1 " (see slide 126), it also needs $E$-square marker symbols "u", " $v$ ", " $w$ ", " "x", " $y$ ", and "z".
- On page 152, Turing specifies a skeleton table for one last $m$-function "con". This is used to identify - and mark with a specified symbol a (e.g. "x") the configuration closest to the right of the current scanned square. See pp. 152-3 of Petzold for explanation and an example.

4. Next (p. 155), it compares the configuration marked " $x$ " (in the list of "instructions" at the start of the tape) with the configuration marked " $y$ " (in the complete configuration at the end of the tape).
5. It loops on like this until a match is found ("Taking the long view, ...", p. 156). Then the operations to be carried out are marked with " $u$ ", and the final state of the instruction with " $y$ " (" $z$ "s are erased).
6. The last complete configuration (at the right of the tape) is marked out (p.157).
7. The figure " 0 " or " 1 " is printed at the end of the tape, if such a print instruction is present (p. 159).
8. The resulting complete configuration is written at the end of the tape, "carrying out the marked instructions" (p. 160). Back to the big loop!

## Alan Turing on Computability and Intelligence

## Lecture 5

Settling Hilbert's Entscheidungsproblem, and the Halting Problem

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy
Hertford College, Oxford


## § 8 Application of the Diagonal Process

Recall from § 5 that Turing has shown that the computable numbers are enumerable.

- But he now remarks that Cantor's diagonal argument might seem to prove otherwise.
- If the computable numbers are enumerable, then we should be able to form a list that contains them all (they are all, of course, endless binary fractions of " 0 "s and " 1 " $s$ ).


## What Prevents Computation of $\beta$ ?

E Yet how can $\beta$ fail to be computable? If we go through the integers one after another, it is relatively easy to identity those that are description numbers of Turing machines and then to construct the standard description.

- Can't we just do this repeatedly, then mimic in turn the $1^{\text {st }}, 2^{\text {nd }}, \ldots n^{\text {th }} \ldots$ machine until we get the $n^{\text {th }}$ digit, swap " 0 " for " 1 " and continue?
- Why won't this give a way of computing $\beta$, and thus refute Turing's claim of enumerability?


## Where the Attempt Fails

E Turing remarks that the reader might feel that "there must be something wrong" with this argument. He then explains where the attempt to construct $\beta$ fails (pp.179-83).

- It breaks down when our would-be machine comes to check its own description number $N$. For then in order to determine the $N^{\text {th }}$ digit that it is supposed to output, it must first discover what the $N^{\text {th }}$ digit of machine number $N$ (i.e. $\mathscr{H}$ itself) would be. So it never finds the answer.


## Turing on Computability and Intelligence


#### Abstract

Another "Diagonal" Argument - Turing's argument here is reminiscent both of Cantor's diagonal arguments, and also Gödel's proof (which involves a formula that indirectly refers to its own unprovability). - He has shown that no Turing machine can be made to predict with certainty whether a given Turing machine, will, or will not, be "circle-free". - Using a conventional programming language, we can prove a related result of more general importance, and with surprising ease ...


## The Halting Problem

E Suppose we have the text of a computer program $P$ (in a standard programming language), taking input from a text file, $T$.

- We would like to have a testing procedure $H$ which will examine $P$ and $T$, and then reliably work out the answer to this question:
- Will program $P$, when run with input $T$, eventually halt, or will it never terminate?
- Turing's argument implies that $H$ is impossible (though of course his own machines never halt if "satisfactory" - see Petzold pp. 328-9).
- Suppose we have such a procedure $H$ :
Text of
program $P$
Text of
input $T$ print("NO")
- We now modify that program: repeat


141

The Contradictory Program Q


- Does program $Q$ halt when given itself as input? If it does, then it doesn't, and if it doesn't, then it does! So $Q$ is an impossible program, and therefore $H$ must have been impossible too.


## An Important Lemma

E Having proved that no machine can provide a reliable circle-free test, Turing goes on to prove a lemma that will be used later:
"We can show further that there can be no machine $\mathscr{E}$ which, when supplied with the S.D of an arbitrary machine $\mathscr{M}$, will determine whether $\mathscr{M}$ ever prints a given symbol (0 say)." (p. 183)

- The proof follows the familiar pattern, showing that if $\mathscr{E}$ were possible, then a diagonally impossible machine (namely, a circle-free testing machine) would also be possible.


## Turing on Computability and Intelligence

- Suppose we have our 0-testing machine $\mathscr{E}$, to be applied to some arbitrary machine $\mathscr{M}$.
- We then consider machines $\mathscr{M}_{1}, \mathscr{M}_{2}, \ldots \mathscr{M}_{n}, \ldots$, which are just like $\mathscr{M}$ except that they print out $n$ fewer " 0 "s (by replacing the first $n$ " 0 " $s$ with another symbol, e.g. " $\phi$ " - this can be done mechanically by adding more states).
E Now we create another machine $\mathscr{G}$ from $\mathscr{E}$, which operates by mechanically testing $\mathscr{M}_{1} \mathscr{M}_{1}$, $\mathscr{M}_{2}, \ldots$ in turn and outputting a " 0 " each time if and only if $\mathscr{E}$ would decide that the tested $\mathscr{M}$-machine would never generate a " 0 ".
So, for example, if $\mathscr{M}$ would print 4 " 0 "s, we get: 145

145

## Deducing the Lemma

E So if machine $\mathscr{E}$ exists, it follows that we can create a general process to determine whether any machine $\mathscr{M}$ prints " 0 " infinitely often. Clearly exactly the same process can be followed for "1" (p. 186).

- But then, "By a combination of these processes we have a process for determining whether $\mathscr{I}$ prints an infinity of figures, i.e. we have a process for determining whether $\mathscr{M}$ is circlefree. There can therefore be no machine $\mathscr{E}$."


## Turing's First Argument for the Church-Turing Thesis

- Turing ( § 9, p. 190) proposes three types of arguments for his claim about the generality of Turing machines (the third involves outlining "large classes of numbers which are computable", and constitutes § 10 - see slide 157 below).
- The first type of argument (pages190-4) maintains that a Turing machine can compute any number that would be computable by a human following a definite process. This was anticipated back in § 1, at p. 68 (see Lecture 3, slides 71-7).


## § 9 The Extent of the Computable Numbers

- § 9 is devoted to showing "that the 'computable' numbers include all numbers which would naturally be regarded as computable".
- Generalised, this is now widely known as:


## The Church-Turing Thesis

Any effectively calculable function can be computed by a Turing machine (or an equivalent process).

| Machine | output 1 | output 2 | output 3 | output 4 | output 5 | $\mathscr{G}$ outputs |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\mathscr{M}$ | 0 | 0 | 0 | 0 | - | - |
| $\mathscr{M}_{1}$ | $\phi$ | 0 | 0 | 0 | - | - |
| $\mathscr{M}_{2}$ | $\phi$ | $\phi$ | 0 | 0 | - | - |
| $\mathscr{M}_{3}$ | $\phi$ | $\phi$ | $\phi$ | 0 | - | - |
| $\mathscr{M}_{4}$ | $\phi$ | $\phi$ | $\phi$ | $\phi$ | - |  |
| $\mathscr{M}_{5}$ | $\phi$ | $\phi$ | $\phi$ | $\phi$ | - | 0 |

E $\mathscr{G}$ will print " 0 " an infinite number of times, unless $\mathscr{M}$ prints " 0 " infinitely often, in which case $\mathscr{G}$ will never print "0".

- We now test $\mathscr{G}$ itself using $\mathscr{E}$, to find out whether $\mathscr{G}$ ever prints " 0 ". This yields a test whether or not $\mathscr{M}$ prints " 0 " infinitely often.


## Turing's Second Argument

E Turing's second type of argument (if not the particular variant he gives in this paper) has almost certainly been the most influential.

- Here he first argues (pp. 221-8) that "If the notation of the Hilbert functional calculus [i.e. first-order predicate logic] is modified so as to be systematic, and so as to involve only a finite number of symbols, it becomes possible to construct an automatic machine, $\mathscr{K}$, which will find all the provable formulae of the calculus."


## Turing on Computability and Intelligence

## A "British Museum Algorithm"

- Turing sketches how a Turing machine can be defined to generate in turn all provable formulae of a system of axioms expressed in predicate formulae.
- As Petzold explains (pp. 220-1), this is known as a "British Museum" algorithm.

E Turing then briefly points out (p. 229) that any Turing machine can be defined in terms of predicate formulae. (This is demonstrated in far more detail in § 11.)

## Turing is "Scooped" by Church

E In early April 1936, six weeks before Turing submitted his own paper (on May 28), Alonzo Church of Princeton sent his two-page "A Note on the Entscheidungsproblem" for publication in Volume 1 of The Journal of Symbolic Logic.

- Church had proved, using his "Lambda Calculus", that the Entscheidungsproblem is unsolvable.
- Turing's paper was considered to be of sufficient interest and novelty to be worth publishing nonetheless, but he was asked to add an Appendix.


## Turing's Appendix

- Turing's Appendix proves in outline "The theorem that all effectively calculable ( $\lambda$-definable) sequences are computable [by a Turing machine] and its converse" (p. 290)
- See Petzold Chapter 15 for much more on the Lambda Calculus etc.; also pp. 325-31.
- In Volume 2 of The Journal of Symbolic Logic (1937), Turing's "Computability and $\lambda$-definability" provided a more rigorous proof. The new paper starts as follows:


## An Important Equivalence

E Thus Turing has argued, in effect:

- that anything computable through predicate logic can be computed by a Turing machine;
- "that the numbers ... definable [in terms of predicate logic] by the use of axioms include all the [Turing-machine] computable numbers." (p. 229)
- So he has thereby sketched "A proof of the equivalence of two definitions" of computable numbers, as he anticipated at p. 190.
"Several definitions have been given to express an exact meaning corresponding to the intuitive idea of 'effective calculability' as applied for instance to functions of positive integers. The purpose of the present paper is to show that the computable functions introduced by the author are identical with the $\lambda$-definable functions of Church and the general recursive functions due to Herbrand and Gödel and developed by Kleene. It is shown [below] that every $\lambda$-definable function is computable and that every computable function is general recursive." (Petzold, p. 298)

[^1]
## Assessing the Church-Turing Thesis

- It seems hard in principle - and maybe even philosophically dubious - to attempt to prove a thesis which identifies an informal notion (i.e. "effective calculability") with a formal notion (e.g. "Turing-machine computability").

But it is striking that all plausible attempts to gine a formal precisification of the informal

156
156

## Turing on Computability and Intelligence

## § 10 Examples of Large Classes of Numbers which are Computable

- § 10 presents a range of results, including:
- various combinations of computable functions must be computable (pp. 240-1, some are proved at 247-56);
- the root of a computable function that crosses zero is computable ( p .244 ), as is the limit of any "computably convergent sequence" (p. 246)
- "The sum of a power series whose coefficients form a computable sequence is a computable function in the interior of its interval of convergence." (p. 247)
$-\pi$ and $e$ are computable (p. 247);
- "all real algebraic numbers are computable" (p. 247)


## § 11 Application to the Entscheidungsproblem

E § 11 finally comes round to Hilbert's famous Entscheidungsproblem, the inspiration for the entire paper (even though the invention of the Turing machine has since been seen as of massive independent significance).Turing proves that this is insoluble using the earlier lemma, and showing that for any machine $\mathscr{M}$, it is possible to construct a formula of predicate logic that is equivalent to the statement that $\mathscr{M}$ at some point prints an " 0 ".

- Turing refers to Hilbert's presentation of "the restricted functional calculus" (i.e. what we call predicate logic without identity). Turing then presents his goal as follows:
"I propose, therefore, to show that there can be no general process for determining whether a given formula $\mathscr{U}$ of the functional calculus $\mathbf{K}$ is provable, i.e. that there can be no machine which, supplied with any one $\mathscr{U}$ of these formulae, will eventually say whether $\mathscr{\mathscr { }}$ is provable." (p. 260)
- He explains (pp. 261-2) that this result is quite different from Gödel's, as we saw in the second lecture (slide 65).


## From Machines to Predicate Logic

- As Turing remarks (p. 262), the proof appears "somewhat lengthy [but] The underlying ideas are quite straightforward."
- As illustrated in the next slide, he defines predicates adequate to describe the configuration and behaviour of Turing machines (pp. 263-7).
E Using these predicates (or "propositional functions"), he defines, given any machine $\mathscr{M}$, his key formula Un $(\mathscr{M})$, expressing the implication from that machine's specification to the consequence that a zero will be printed (p. 267). By the earlier lemma (slides 144-7), this will in general be undecidable.


## Expressing Turing Machine Rules

These predicates enable us to specify any complete configuration of a Turing machine, in terms of the step number (or "cycle") of that configuration.

- The Turing Machine rules (i.e. quintuples or machine table) specify what should happen in moving from one cycle to the next. The next few slides explain how Turing represents this (and can usefully be read together with p. 265-6). To do so, he also needs a predicate for numerical succession (which he had introduced earlier at p. 225, in preparation for discussing Peano's axioms):

$$
\begin{aligned}
& F(x, y) \\
& 162 \quad y \text { is the immediate successor of } x \text { (i.e. } y=x+1 \text { ). }
\end{aligned}
$$

## Turing on Computability and Intelligence

Suppose our machine includes quintuple $q_{i} S_{i} S_{k} L q_{i}$ and that in cycle $x$ of this machine, square $y$ is scanned, in state $q_{i}$ and while containing symbol $S_{j}$ (so the quintuple would apply on that cycle).

- Applying the quintuple, then: in the next cycle $x^{\prime}$, square $y$ will change to symbol $S_{k}$ and the machine will move left (to scan square $y^{\prime}$ ) and into state $q_{l}$. Clearly $x^{\prime}=x+1$ and $y^{\prime}=y-1$, so we have:

$$
F\left(x, x^{\prime}\right) \& F\left(y^{\prime}, y\right)
$$

and for the two cycles a scan/state/symbol formula:

$$
\begin{array}{lccccc}
\text { Cycle } x & I(x, y) & \& & K_{q_{i}}(x) & \& & R_{S_{j}}(x, y) \\
\text { Cycle } x^{\prime} & I\left(x^{\prime}, y^{\prime}\right) & \& & K_{q_{l}}\left(x^{\prime}\right) & \& & R_{S_{k}}\left(x^{\prime}, y\right) \\
& \uparrow & & \uparrow_{\text {Scanned square }} & & \\
& \text { State } & & \uparrow_{\text {Symbol }}
\end{array}
$$

■ The formulae for Inst $\left\{q_{i} S_{j} S_{k} R q_{l}\right\}$ and Inst $\left\{q_{i} S_{j} S_{k} N q_{l}\right\}$ - to deal with quintuples in which the machine moves right or stays on the same square - will differ accordingly.

- Thus Inst $\left\{q_{i} S_{j} S_{k} R q_{l}\right\}$ will be the formula (p. 266):

| $\left(x, y, x^{\prime}, y^{\prime}\right) \quad \forall$ | $\forall x, y, x^{\prime}$, and $y^{\prime}$ |
| :---: | :---: |
| $\left\{I(x, y) \& K_{q_{i}}(x) \& R_{S_{j}}(x, y)\right.$ | IF cycle $x$ scan/state/symbol |
| \& $F\left(x, x^{\prime}\right) \& F\left(y, y^{\prime}\right)$ | \& $x^{\prime}=x+1, y^{\prime}=y+1$ |
| T | THEN |
| $\left(I\left(x^{\prime}, y^{\prime}\right) \& K_{q_{l}}\left(x^{\prime}\right) \& R_{S_{k}}\left(x^{\prime}, y\right)\right.$ | cycle $x^{\prime}$ scan/state/symbol |
| \& $(z)\left[F\left(z, y^{\prime}\right)\right.$ | \& $\forall z$ [either $z=y$ (i.e. $y^{\prime}-1$ ) |
| $\vee\left(\left[R_{S_{0}}(x, z) \rightarrow R_{S_{0}}\left(x^{\prime}, z\right)\right]\right.$ | \& or square $z$ keeps |
| $165 \quad\left[R_{S_{1}}(x, z) \rightarrow R_{S_{1}}\left(x^{\prime}, z\right)\right] \&$ | \& ...])])\} the same symbol] |

165

## From Machine Description To (Undecidable) Zero-Print Statement

- Every quintuple in machine $\mathscr{I}$ can be represented by one of these long formulae - which we have abbreviated as Inst $\left\{q_{i} S_{j} S_{k} N q_{l}\right\}$, Inst $\left\{q_{i} S_{j} S_{k} N q_{l}\right\}$ or Inst $\left\{q_{i} S_{j} S_{k} N q_{l}\right\}$. We then take the conjunction of all these (pp. 267, 311), and call that $\operatorname{Des}(\mathscr{M})$.
- Thus Des $(\mathscr{M})$ gives a complete description, in predicate logic, of the operation of $\mathscr{M}$ 's machine table.

E Following the strategy explained in slide 160 above, Turing next uses $\operatorname{Des}(\mathscr{M})$ to create another formula, representing the statement that $\mathscr{M}$ will at some point print a zero (character $S_{1}$ ) on its tape.

- We also need to express that all squares other than $y$ will contain the same symbol in cycles $x$ and $x^{\prime}$.
- This yields the following formula (which Turing abbreviates as "Inst $\left\{q_{i} S_{j} S_{k} L q_{l}\right\}^{\prime}$, p. 265) to express that quintuple:
( $x, y, x^{\prime}, y^{\prime}$ )
$\forall x, y, x^{\prime}$, and $y$
$\left\{I(x, y) \& K_{q_{i}}(x) \& R_{S_{j}}(x, y) \quad\right.$ IF cycle $x$ scan/state/symbol
$\& F\left(x, x^{\prime}\right) \& F\left(y^{\prime}, y\right)$

$$
\& x^{\prime}=x+1, y^{\prime}=y-1
$$

$\rightarrow$
THEN
$\left(I\left(x^{\prime}, y^{\prime}\right) \& K_{q_{l}}\left(x^{\prime}\right) \& R_{S_{k}}\left(x^{\prime}, y\right) \quad\right.$ cycle $x^{\prime}$ scan/state/symbol
\& $(z)\left[F\left(y^{\prime}, z\right)\right.$
\& $\forall z$ [either $z=y$ (i.e. $y^{\prime}+1$ )

$$
\begin{array}{ll}
\vee\left(\left[R_{S_{0}}(x, z) \rightarrow R_{S_{0}}\left(x^{\prime}, z\right)\right] \&\right. & \text { or square } z \text { keeps } \\
& \left.\left.\left.\left.\left.\left.\left[R_{S_{1}}(x, z) \rightarrow R_{S_{1}}\left(x^{\prime}, z\right)\right] \& \ldots\right]\right)\right]\right)\right\} \quad \text { the same symbol }\right]
\end{array}
$$

164

E Inst $\left\{q_{i} S_{j} S_{k} N q_{l}\right\}$ involves no machine movement, but we still keep $y^{\prime}$ in the formula in order to express the identity of $z$ with $y$ (we cannot write " $z=y$ ", since Hilbert's restricted functional calculus does not include the identity sign):

| $\left(x, y, x^{\prime}, y^{\prime}\right) \quad \forall$ | $\forall x, y, x^{\prime}$, and $y^{\prime}$ |
| :---: | :---: |
| $\left\{I(x, y) \& K_{q_{i}}(x) \& R_{S_{j}}(x, y)\right.$ | IF cycle $x$ scan/state/symbol |
| \& $F\left(x, x^{\prime}\right) \& F\left(y, y^{\prime}\right)$ | \& $x^{\prime}=x+1, y^{\prime}=y+1$ |
| T | THEN |
| $\left(I\left(x^{\prime}, y\right) \& K_{q_{l}}\left(x^{\prime}\right) \& R_{S_{k}}\left(x^{\prime}, y\right)\right.$ | cycle $x^{\prime}$ scan/state/symbol |
| \& $(z)\left[F\left(z, y^{\prime}\right)\right.$ | \& $\forall z$ [either $z=y$ (i.e. $y^{\prime}-1$ ) |
| $\vee\left(\left[R_{S_{0}}(x, z) \rightarrow R_{S_{0}}\left(x^{\prime}, z\right)\right]\right.$ | \& or square $z$ keeps |
| $166 \quad\left[R_{S_{1}}(x, z) \rightarrow R_{S_{1}}\left(x^{\prime}, z\right)\right] \&$ | \& ...])])\} the same symbol] |

166

## Peano's Axioms

- A complication here is that Turing needs to specify the relevant properties of the number sequence (corresponding to the sequence of configurations).
- These properties are captured by Peano's Axioms, of which this is one version (see Petzold p. 223):

1. Zero is a number.
2. Every number has a successor that is also a number.
3. Zero is not the successor to any number.
4. Two numbers that are the successors to the same number are equal.
5. If $P(0)$ is true, and if $P(x)$ implies $P(\operatorname{successor}(x))$ for all numbers $x$, then $P(x)$ is true for all numbers $x$.

## Turing on Computability and Intelligence

Turing's original paper slips up in defining the Peano conditions (pp. 226-7, 267-8). He provided a correction (pp. 268, 311), which uses $Q$ as an abbreviation for:
$(x)(\exists w)(y, z)\left\{\begin{array}{c}F(x, w) \&(F(x, y) \rightarrow G(x, y)) \\ \&(F(x, z) \& G(z, y) \rightarrow G(x, y)) \\ \&\left[\begin{array}{c}G(z, x) \vee(G(x, y) \& F(y, z)) \\ \vee(F(x, y) \& F(z, y)) \rightarrow(\neg F(x, z))\end{array}\right]\end{array}\right\}$
If $x$ is any number, there is some $w$ which is the successor of $x$, and for all $y$ and $z$, if $y$ is a successor of $x$ then $y$ is greater than $x$, and if $z$ is a successor of $x$ and $y$ is greater than $z$ then $y$ is greater than $x$, and if either $x$ is greater than $z$,
or $y$ is greater than $x$ and $z$ is a successor of $y$,
or $y$ succeeds both $x$ and $z$,
then $z$ is not a successor of $x$.
169
(But even this isn't ideal, e.g. it doesn't rule out branching.)

## The Undecidable Formula Un( $\mathscr{M})$

E The formula Un $(\mathscr{I})$ for machine $\mathscr{M}$, which is in general undecidable, is as follows (pp. 268, 311):
$(\exists u)\left[Q \&(y) R_{S_{0}}(u, y) \& I(u, u) \& K_{q_{1}}(u) \& \operatorname{Des}(\mathscr{M})\right]$ $\rightarrow(\exists s)(\exists t) R_{S_{1}}(s, t)$

- Apart from $Q$ and $\operatorname{Des}(\mathscr{A})$, the antecedent of this conditional states that there is some $u$ (namely zero), the number of a configuration (i.e. the initial one) in which: $(R) S_{0}$, the blank, is on every square; (I) square zero is scanned; and $(K)$ the machine is in state $q_{1}$.
- The consequent says that there exist some $s$ and $t$ such that $S_{1}$, the zero symbol, appears on square $t$ of the tape in cycle (i.e. configuration number) $s$.
- To prove Lemma 1, Turing formulates a sequence of formulae $C C_{0}, C C_{1}, C C_{2}$, etc., representing the sequence of complete configurations as $\mathscr{I}$ proceeds (p. 271).
- Here he uses an earlier abbreviation (from p. 268), with "A $(\mathbb{K})$ " standing for:

$$
Q \&(y) R_{S_{0}}(u, y) \& I(u, u) \& K_{q_{1}}(u) \& \operatorname{Des}(\mathscr{M})
$$

- He also uses " $F(n)$ " to abbreviate the numeric successor relations (from 0 up to $n$ inclusive).
- He then shows, by induction, that "all formulae of the form $\mathrm{A}(\mathscr{M}) \& F^{(n)} \rightarrow C C_{n}$ are provable" (p. 272: base case pp. 272-3; induction step pp. 273-5).
- To sum up, the lemma of § 8 proved that no machine can be constructed which will reliably determine whether any specified machine will ever print a " 0 ".
E But since that behaviour (printing a " 0 ", or not doing so) can be captured by a predicate logic formula $\operatorname{Un}(\mathscr{M})$, it follows that no machine can be constructed which will reliably determine whether or not that formula is provable.
- Hence Hilbert's Entscheidungsproblem - to devise a mechanical procedure to determine whether or not any predicate logic formula is provable, has no solution. QED!!


## Alan Turing on Computability and Intelligence

## Lecture 6

"Computing Machinery and Intelligence": Overview of Turing's 1950 Paper

## Peter Millican

Gilbert Ryle Fellow
and Professor of Philosophy
Hertford College, Oxford


## Useful Books

E Books on the Philosophy of Al

- Jack Copeland, Artificial Intelligence: A Philosophical Introduction (Blackwell, 1993).
- Margaret A. Boden (ed.), The Philosophy of Artificial Intelligence (OUP, 1990).
- Douglas R. Hofstadter and Daniel C. Dennett (eds), The Mind's I (Penguin, 1981).
- The Chinese Room Argument
- John Searle, Minds, Brains and Science (Penguin, 1989).


## Useful Collections of Papers

- On the Turing Test
- S. Barry Cooper and Jan van Leeuwen (eds), Alan Turing: His Work and Impact (Elsevier, 2013) - has a range of papers on the Turing Test.
- Peter Millican and Andy Clark (eds), Machines and Thought (OUP, 1996) - the introduction and the first seven papers are relevant.
- On the Chinese Room Argument
- John Preston and Mark Bishop (eds), Views into the Chinese Room (OUP, 2002).


## Useful Web Resources

- Stanford Encyclopedia of Philosophy
- At http://plato.stanford.edu/ - see articles on: E "The TuringTest"
E"The Chinese Room Argument"
- Web Resources on the Turing Test
- On Andrew Hodges' "Alan Turing" website, the paper "Alan Turing and the Turing Test": http://www.turing.org.uk/publications/testbook.html
- Ayse Pinar Saygin, Ilyas Cicekli \& Varol Akman, "Turing Test: 50 Years Later": http://www.cs.bilkent. edu.tr/~akman/jour-papers/mam/mam2000.pdf

177
178

## § 1 The Imitation Game

- Turing starts his paper a bit confusingly:
- "I propose to consider the question, 'Can machines think?' ..."
- "If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used ... the answer ... is to be sought in a statistical survey. But this is absurd. Instead ... I shall replace the question by another, which is closely related to it ... [but] relatively unambiguous."


## Turing on Computability and Intelligence

## The Gender Imitation Game

- Turing's replacement question is set in the context of an "imitation game", which is first introduced in a form requiring an interrogator, $C$, to guess the gender of two participants (a man, $A$, and a woman, $B$ ), based on the answers they give to C's questions.
- The man, $A$, attempts by his answers to convince $C$ that he is the woman and $B$ is the man. The woman, $B$, attempts to convince $C$ that she is indeed the woman.



## Constraints on the Game

The interrogator obviously needs to be in a different room from the participants, so as to be unable to see them. But further:

- "In order that tones of voice may not help the interrogator the answers should be written, or better still, typewritten. The ideal arrangement is to have a teleprinter communicating between the two rooms."
- Communicating text by teleprinter also makes the game easily extendable to a computer ...


## The Computer Imitation Game

E "We now ask ..., 'What will happen when a machine takes the place of $A$ (the deceitful man) in this game?' Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, 'Can machines think?"'

- The relationship between these questions and the original is left obscure. And it's not clear (until § 2 and §5) that the computer's role is to imitate a human ("man") rather than specifically a woman.


## § 2 Critique of the New Problem <br> - § 2 helps to clarify Turing's idea: the interrogator's questions can be used to elicit the computer's "knowledge" about "almost any ... of the fields of human endeavour", e.g. poetry composition, arithmetic, or chess. <br> - The teleprinter setup "has the advantage of drawing a fairly sharp line between the physical and the intellectual capacities of a man". In the paper "it will be assumed that the best strategy is to try to provide answers that would naturally be given by a man".

## Turing on Computability and Intelligence

## Flexibility of the Q\&A Format

Q: Please write me a sonnet on the subject of the Forth Bridge.
A: Count me out on this one. I never could write poetry.
Q: Add 34957 to 70764 .
A: (Pause about 30 seconds and then give as answer) 105621.

Q: Do you play chess?
A: Yes.
Q: I have K at my K1, and no other pieces. You have only K at K6 and R at R1. It is your move. What do you play? A: (After a pause of 15 seconds) $R$-R8 mate.

187

## Bias in Favour of Human Thought?

E But Turing notes the possible objection: "May not machines carry out something which ought to be described as thinking but which is very different from what a [human person] does?"

- He responds: "This objection is a very strong one, but at least we can say that, if, nevertheless, a machine can be constructed to play the imitation game satisfactorily, we need not be troubled by this objection."
- This seems to be suggesting that success in the imitation game is sufficient to prove intelligence, but not a necessary condition. spring day" do as well or better?
Witness: It wouldn't scan.
How about "a winter's day"? That would scan all right.
Yes, but nobody wants to be compared to a winter's day.
Would you say Mr. Pickwick reminded you of Christmas? In a way.
Yet Christmas is a winter's day, and I do not think Mr. Pickwick would mind the comparison.

I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

## §4 Digital Computers

E "The idea behind digital computers [is] that these machines are intended to carry out any operations which could be done by a human computer." - here we see clear echoes of the 1936 paper.

- Turing then gives an outline of how a digital computer might operate, remarking that "they can in fact mimic the actions of a human computer very closely".


## Turing on Computability and Intelligence

Varieties of Digital Computer<br>- Turing considers some particular cases:<br>- A "digital computer with a random element", which can be simulated by including a pseudo-random process (e.g. using digits of $\pi$ ).<br>- A "computer with an unlimited store"; "Such computers have special theoretical interest and will be called infinitive capacity computers".<br>- He ascribes the idea of a digital computer to Charles Babbage, thus illustrating that they can be mechanical and need not be electrical.

## The Programming Question

- Because digital computers are universal, the Imitation Game question reduces to:
- "Let us fix our attention on one particular digital computer $C$. Is it true that by modifying this computer to have an adequate storage, suitably increasing its speed of action, and providing it with an appropriate programme, $C$ can be made to play satisfactorily the part of $A$ in the imitation game, the part of $B$ being taken by a man?"
- Apparently $C$ is supposed to be imitating a human, rather than playing a gender-related game.

Before considering the objections, however, Turing offers his own predictions:

- "I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about $10^{9}$, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent. chance of making the right identification after five minutes of questioning."
- "The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."


## (1) The Theological Objection

"Thinking is a function of man's immortal soul. God has given an immortal soul to every man and woman, but not to any other animal or to machines. Hence no animal or machine can think."

- Turing suggests it would be more persuasive to rank animals with people, and criticises the arbitrariness of religious views (e.g. that women don't have souls).
- But suppose there are souls given by God - why shouldn't He give one to a computer?
- Turing alludes to Galileo: as knowledge advances, religious dogmas can come to seem obsolete.


## Turing on Computability and Intelligence

## (2) The "Heads in the Sand" Objection

"The consequences of machines thinking would be too dreadful. Let us hope and believe that they cannot do so."

- Turing suggests that this concern, though "seldom expressed quite so openly", often motivates those who are opposed to the idea of machine intelligence.
- We like to think that we are "superior to the rest of creation", and "intellectual people ... value the power of thinking more highly than others".
- "Consolation" is more appropriate than refutation: "perhaps ... in the transmigration of souls." Here Turing seems to be having fun, as Gandy suggested.


## (4) The Argument from Consciousness

- Turing quotes from Geoffrey Jefferson's 1949 "Lister Oration":
- "Not until a machine can write a sonnet or compose a concerto because of thoughts and emotions felt, and not by the chance fall of symbols, could we agree that machine equals brain - that is, not only write it but know that it had written it."
- "No mechanism could feel (and not merely artificially signal, an easy contrivance) pleasure at its successes, grief when its valves fuse, be made miserable ..., be charmed ..., be angry or depressed ..."
201


## 202

## Turing's Response to Jefferson

E Turing's answer to Jefferson starts with some amusing (but dubious) rhetoric:
"This argument appears to be a denial of the validity of our test. According to the most extreme form of this view ... the only way to know that [either a machine or] a man thinks is to be that particular man. It is in fact the solipsist point of view. It may be the most logical view to hold but it makes communication of ideas difficult. A is liable to believe 'A thinks but B does not' ... Instead of arguing continually over this point it is usual to have the polite convention that everyone thinks."

202

Interrogator: In the first line of your sonnet which reads "Shall I compare thee to a summer's day", would not "a spring day" do as well or better?
Witness: It wouldn't scan.
How about "a winter's day"? That would scan all right.
Yes, but nobody wants to be compared to a winter's day.
Would you say Mr. Pickwick reminded you of Christmas? In a way.

Yet Christmas is a winter's day, and I do not think Mr. Pickwick would mind the comparison.
I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

## Turing on Computability and Intelligence

## Two Distinct Lines of Thought

E There are two quite different lines of thought here, which Turing would have done well to distinguish:

- Jefferson is "denying the validity" of the Turing test because it does not test for genuine consciousness, and genuine consciousness (rather than "artificial signalling") is necessary for intelligence.
- "Artificial signalling" of apparent emotions is unworthy of being deemed intelligent because it is an "easy contrivance".


## A Better Response

- Turing's response to the second point (on the alleged "easy contrivance") is much stronger than his response to the first (regarding consciousness, solipsism etc.).
- He would have been better to say, after giving the sonnet example (and others):
- "if the answers were as satisfactory and sustained as in the above passage ... then there would be reason to call the machine 'intelligent' irrespective of whether or not it has genuine feelings. Intelligence need not require consciousness."


## (5) Arguments from Various Disabilities

- A machine could not, allegedly, be kind, beautiful, friendly, have initiative, have a sense of humour, tell right from wrong, make mistakes, fall in love, enjoy strawberries and cream, learn from experience, use words properly, be the subject of its own thought, do something really new.
E The limited machines of 1950 could not do these, but it requires argument to show than none could.
Several of these alleged limits seem just to take us back to the argument from consciousness.

207
208

- To be fair to Ada Lovelace, we should note her visionary anticipation of computer creativity in her note A on Babbage's Analytical Engine:
"The operating mechanism ... might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."
https://www.fourmilab.ch/babbage/sketch.html


# Turing on Computability and Intelligence 

## (7) Argument from Continuity in the Nervous System

"The nervous system is certainly not a discretestate machine. A small error in the information about the size of a nervous impulse impinging on a neuron, may make a large difference to the size of the outgoing impulse."

- In response, Turing points out that a discretestate machine could mimic a continuous system sufficiently closely that a human would find it extremely hard to tell the difference.


## (8) The Argument from Informality of Behaviour

- Turing reduces this argument to:
"If each man had a definite set of rules of conduct by which he regulated his life he would be no better than a machine. But there are no such rules, so men cannot be machines."
- He points out that formally it is a fallacy $(R \rightarrow M$ is not the same as $M \rightarrow R)$.
- Moreover it is hard to establish that we are not in fact governed by "laws of behaviour" (to be distinguished from "rules of conduct")


## (9) The Argument from

 Extra-Sensory PerceptionE Turing's sympathy towards telepathy, clairvoyance, precognition and psycho-kinesis might seem surprising: "the statistical evidence, at least for telepathy, is overwhelming".

- Few would now agree with this, but Turing was apparently impressed with the work of J. B. Rhine (see Hodges' biography, p. 416).
- Turing seems also to assume that if extra-sensory perception were possible, no machine could mimic this, so we might need a "telepathy-proof room". 213


## § 7 Learning Machines

E Turing admits at the beginning of § 7 that

- "I have no very convincing arguments of a positive nature to support my views."
E Since his view is centred on performance in the imitation game (i.e. the "Turing test"),
- "The only really satisfactory support that can be given for the view expressed at the beginning of $\S 6$, will be that provided by waiting for the end of the century and then doing the experiment."

214

After a discussion of technological needs (storage of around $10^{9}$, speed around 1,000 times faster than nerve cells), he focuses on the problem of programming a machine to play the game successfully.
The solution, he suggests, may be to try to simulate a baby's mind rather than an adult's, and provide it with the ability to learn.

- Here Turing's discussion seems quite unrealistic, but he did not have the advantage of our further 60 years of experience, which have shown how difficult it is to implement general learning systems to compete with the products of evolution.

E One important point Turing makes here is that a learning machine is highly likely to behave in ways that its programmers could neither foresee nor understand, and also to make "mistakes". Trial and error learning, in particular, requires some element of randomness.
E "Many people think that a very abstract activity, like the playing of chess" might be a good place to start in attempting to match human intelligence.
E But perhaps instead "it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English" so that it could then "follow the normal teaching of a child."

- On this uncertain note, the paper ends.

216
216

## Alan Turing on Computability and Intelligence

## Lecture 7

Blockhead, the Chinese
Room, and ELIZA

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy

Hertford College, Oxford


## BlockHead

In 1981, Ned Block attacked the Turing Test on the grounds that it can be passed by a mindless machine.

- His postulated machine, widely known as "Blockhead", works by storing every possible "sensible" conversation of a given length, and choosing its responses accordingly.
- Something working so mindlessly, by a mechanism that involves no understanding, surely cannot be called intelligent, but it will always generate sensible conversation!


## An Important Distinction

Imagine a system that exhibits some impressive behaviour. In attributing intelligence to it, we could mean either:

- That behaviour is definitive of intelligence: anything that behaves like that is ipso facto correctly described as "intelligent".
- That behaviour provides strong evidence of intelligence, because it could only plausibly be generated by something with the capacity for sophisticated information processing (etc.).

Consider the following thought-experiment:
"suppose that someone were to write a computer program of only around 50 lines of code (in a standard general programming language), which could play chess at a grandmaster level in real time. Such a crude program could not possibly count as genuinely intelligent. Hence grandmaster performance at chess is not a reliable proof even of intelligent chess-playing."

- The same argument can be made about any domain, so apparently we can never take expert performance as proving intelligence!
- But like Blockhead, this scenario isn't remotely plausible, so we should refuse to be persuaded.


## Turing on Computability and Intelligence

## Thought-Experiments as "Intuition Pumps"

"If you look at the history of philosophy, you see that all the great and influential stuff has been technically full of holes but utterly memorable and vivid. They are what I call 'intuition pumps' - lovely thought experiments. Like Plato's cave, and Descartes's evil demon, and Hobbes' vision of the state of nature and the social contract, and even Kant's idea of the categorical imperative. I don't know of any philosopher who thinks any one of those is a logically sound argument for anything. But they're wonderful imagination grabbers, jungle gyms for the imagination. They structure the way you think about a problem. These are the real legacy of the history of philosophy."

Daniel Dennett, in The Third Culture, ed. Brockman

But the man inside the room has no knowledge whatever of the Chinese language or of the semantics - the meaning or significance - of the symbols he is reading or writing.

- Instead, he is generating his written "answers" by strictly applying rules based purely on the syntax - the shape and structure - of the "question" character strings that he receives, these rules being specified in books contained within the room. Searle gives an example:
"Take a squiggle-squiggle sign out of basket number one and put it next to a squoggle-squoggle sign from basket number two." (Searle 1984, p. 32)


228

## Chinese Variations

- In the original version of the Chinese Room (in "Minds, Brains, and Programs", Behavioral and Brain Sciences 3, 1980, pp. 417-24), questions are limited to testing the man's comprehension of a fixed story written in Chinese.

E In the more famous later version (in Minds, Brains \& Science, 1984, p. 32), there is no such restriction: the questions can apparently be as varied as those in the Turing Test.

- This takes the required processing to a whole new level of sophistication (and implausibility!).


## Searle's Conclusion

Clearly the man in the room does not understand Chinese, despite the fact that he is generating meaningful replies.

Searle draws the moral that:

- "Understanding a language, or indeed having mental states at all, involves more than just having a bunch of formal symbols. It involves having an interpretation, or a meaning attached to those symbols. [Computer] programs [like the rules followed by the man in the room] are purely formally specifiable - that is, they have no semantic content." (1984, p. 33)


## What Exactly is Searle Denying?

Most of the time, Searle expresses his thesis as a denial of "intentionality" or "semantic content".
E But he also denies that digital machines can have "a mind", "mental states", "mental content", "cognitive states", or "cognitive processes".

- And he describes his argument as attacking the claim of "strong artificial intelligence", that digital machines can "think" or have "consciousness".
(For examples of these phrases, see Searle 1980, p. 417; 1984, pp. 36-7; 2002 § I, p. 56.)
231



## The System Reply

- Many argue that, though the man in the room does not understand Chinese, the system of which he is a part does (as shown by its intelligent responses).
- Searle rebuts this as follows:
"They argue that it is the whole system, including the room, the baskets full of symbols and the ledgers containing the programs and perhaps other items as well, taken as a totality, that understand Chinese. But this is subject to exactly the same objection... There is no way that the system can get from the syntax to the semantics. I, as the central processing unit have no way of figuring out what any of these symbols means; but then neither does the whole system." (1984, p. 34)


## The Cautious Interpretation

E Searle is often interpreted as arguing that "machines cannot think", but he accepts that we are thinking "machines" (in the sense of physical things working according to natural laws).

E When most cautious, he seems to be objecting to the idea that digital computers can have semantic (i.e. meaningful, genuinely intentional) states purely in virtue of following a symbolic algorithm.

- But he often seems to go significantly beyond this cautious interpretation (e.g. when addressing the "robot reply", which we consider shortly).


## Copeland Presses the Attack

- Copeland (1993, §6.2) argues strongly that Searle's rebuttal begs the question:
- As a matter of logic, the man's lack of understanding does not prove that the system of which he is a part does not understand Chinese. (Compare the silly argument: "Bill the cleaner has never sold pyjamas to Korea; therefore Bill's company has never sold ...")
- If Searle aims to prove that symbol manipulation cannot produce understanding, he cannot appeal to this thesis to defend himself against the System Reply.
- The Chinese Room might retain force as an "intuition pump", but it provides little force of argument here.


# Turing on Computability and Intelligence 

## The Robot Reply

- Note again Searle's main point, that the symbol processing performed by the man (or computer) in the room involves no semantic content or understanding of what the symbols signify.
- A tempting response is to suggest that such content could be given if the system were embedded in the world: if it were responsive to physical sensors, and could cause a robot to act.
- Searle replies that the man in the room has no understanding of any such inputs and outputs ...


#### Abstract

"As long as we suppose that the robot has only a computer for a brain then, even though it might behave exactly as if it understood Chinese, it would still have no way of getting from the syntax to the semantics of Chinese. You can see this if you imagine that I am the computer. Inside a room in the robot's skull I shuffle symbols without knowing that some of them come in to me from television cameras attached to the robot's head and others go out to move the robot's arms and legs. As long as all I have is a formal computer program, I have no way of attaching any meaning to any of the symbols."


(Searle 1984, pp. 34-5)

## Searle versus Turing

- We'll return to this discussion in the last lecture, but for now, notice the intimate contrariety between Searle's Chinese Room and the Turing test:
- Both thought-experiments postulate an algorithmic system capable of generating conversation that is indistinguishable in quality from that of an intelligent native speaker;
- But they draw opposite conclusions, by focusing on different aspects of the situation:


## Back to Reality

As in the case of Blockhead, we can inject some realism by asking how plausible the two thought-experiments are in practice.

- The Chinese Room (especially in its later unrestricted version) is as wildly implausible as Blockhead, supposing sophisticated linguistic behaviour generated in real time by manually consulting books of rules contained within a room (with no scope for sensory input, real-time updating, or emotional reaction etc.).
- Turing's predictions seem more realistic ...


## Turing's Predictions from § 6 <br> E "I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about $10^{9}$, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent. chance of making the right identification after five minutes of questioning." <br> E "The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."

## Turing on Computability and Intelligence

## A Correct Prediction? <br> E So by 2000 "one will be able to speak of machines thinking without expecting to be contradicted". And indeed by 2000 this conversation was plausible: <br> "Why is the computer taking so long to respond?" <br> "It's thinking hard, because it's realized that if it tries to defend against my attack by bringing its knight over to protect the king, l'll be able to grab its pawn on the other side. It's displaying now that it assesses the position as better for me materially, but it's predicting that it won't be too badly off if it decides to let the pawn fall, so I think it must be expecting to get some activity to compensate."

Another Plausible Prediction, but ...

E Turing's other prediction is that by 2000:
"... it will be possible to programme computers ... to make them play the imitation game so well that an average interrogator will not have more than 70 per cent. chance of making the right identification after five minutes of questioning."
This again looks plausible in retrospect. If sustained and systematic research effort had been directed towards this goal, it seems quite likely that it could have been achieved by 2000.

- However the prediction is plausible for a bad reason!


## TURING TEST SUCCESS MARKS MILESTONE IN COMPUTING HISTORY

Release Date 08 June 2014
An historic milestone in artificial intelligence set by Alan
Turing the father of modern computer science has been
achieved at at event organised by the University of Read ding.
The 65 year-old iconic Turing Test was passed for the very
first time by computer programme Eugene Goostman
during Turing Test 2014 held at the renowned Royal Society
in London on Saturday.
Eugene simulates a 13 year old boy and was developed in

244


246

## Turing on Computability and Intelligence



247

## Elizabeth: ELIZA's Cleverer Sister

- Elizabeth is an educational chatbot-creation system, an adaptation of ELIZA in which the various selection, substitution, and phrase storage mechanisms have been enhanced and generalised to increase both its flexibility and its potential adaptability.
- The system also incorporates analysis tables to show what processing has taken place, thus providing a learning tool.
- Elizabeth can be downloaded from:
www.philocomp.net/ai/elizabeth


## Simple Patterns and Responses

■ "... alike ..."
"IN WHAT WAY?"
"something or other"
"My X"
[ "I need $Y$ "
■ "... my mother"
"FOR EXAMPLE?"
"Your X?"
"WHAT IF YOU GOT Y?"
"... YOUR FAMILY ..."
"Who ELSE ...?"
"Does that have ANYTHING TO DO WITH Your X?"

248
248

## Chatbots are NOT Intelligent!

E Experimentation with chatbots (e.g. the ELIZA script available within Elizabeth) can quickly strengthen one's impression that Searle is right: there is no semantics here, only blind manipulation of syntax.

E The script that runs "Eugene Goostman", the chatbot that supposedly "passed the Turing Test" in 2014, follows the standard ELIZA-style pattern. Faced with focused questioning, it soon reveals its stupidity ..

## 250

[^2]
## Turing on Computability and Intelligence

## Uses for Chatbots <br> Chatbots (or chatterbots) can be genuinely useful learning tools, one lesson being how quickly one discovers their limitations!to interpret their output as "intelligent" : <br> - Most people prefer to interact conversationally. <br> - Automated "help" systems can be useful even if their responses are pre-packaged. <br> - Chatbot methods can often succeed in eliciting and identifying relevant information quickly.



254

## Undiscriminating Human Judgement

- Turing acknowledged that passing his Test wasn't a necessary condition for intelligence.
But unless interpreted fairly rigorously, it's not sufficient either, because it's too easy to pass due to human lack of critical discernment!
- Much of our conversation is sloppy and careless;
- Hence we too easily interpret sloppy and careless conversation as indicative of intelligence.
- It's a shame Turing gave the impression that "better" performance in his Test gives a useful criterion of relative intelligence - it doesn't!


## Objectionable Anthropocentrism

As we saw before (slide 176), Turing himself acknowledges that a machine might "think" in ways very different from a human.

- Moreover mimicking "subcognitive" human responses is likely to be extremely difficult.
- Robert French (1990) suggests questions like:
"Rate 'Flugly' as the name of a glamorous model or a cuddly toy."
- But testing for indistinguishability of this sort - in tastes and instinctive reactions - seems inappropriate when testing for intelligence.

256

## The Tutoring Test

Perhaps a better form of test is one focused on revelation of information processing - in a teaching task - rather than deception.

- Enables focus on a specific domain, e.g. some aspect of chemistry, which is suitably deep with complex informational structures.
- No need to pretend, and no expectation of indistinguishability: the test is how well the system can teach the theory (e.g as well as a human?).
- Could provide seriously useful products, and also stimulate progress towards "intelligent" processing.


## Alan Turing on Computability and Intelligence

Lecture 8
Searle versus Turing: Conclusion

Peter Millican
Gilbert Ryle Fellow and Professor of Philosophy
Hertford College, Oxford


## Summary on Turing So Far

- Turing's 50-year predictions in his 1950 paper look plausible (slides 240-2).
- However as a criterion of intelligence, his proposed Turing Test is very dubious:
- If interpreted generously (e.g. $\leq 70 \%$ success by an average interrogator after 5 minutes), it is undermined by the discovery that humans are easy to fool by chatbots (slides 243-52).
- If interpreted more strictly, it is too demanding: an intelligent system need not be designed to mimic human cultural reactions etc. (slides 255-7).


## An "Intelligent" Tutoring System?

E Imagine a computerised tutoring system which is capable of highly sophisticated information processing in a complex domain (slide 258).

- It operates not by pretence, but rather by genuine responsiveness to information structures, in ways that reflect how human experts would think.
- Nevertheless, Searle will insist that the system's processing is merely "syntactic", with no genuine "semantics".
- It has no real understanding of the domain - its

262

261
262

## Two Responses to Searle

- First, we might accept that in our imagined tutoring system the processed information has no intrinsic, system-understood, semantics.
- But we might nevertheless consider the processing of the information to be "intelligent".
- Searle seems to assume that intelligence requires intrinsic "semantics", but this could be challenged.
- Second, we can consider ways in which a computer system might perhaps achieve intrinsic "semantics" ...


## Arithmetic and Abstract Games

- Searle's objection seems quite strong in respect of information processing about physical things (e.g. chemicals or trees): "surely", we are inclined to say, "mere internal processing of formal symbols cannot genuinely constitute thought about real trees!"

E But this isn't so clear with thought about abstract entities such as numbers or chess positions, where apparently "intelligent" processing need not be responsive to the properties of concrete things like trees, but only to appropriate logical relations, which can apparently be represented formally.

## It's a Lumberjack, and It's OK?

- Imagine a robotic crane, armed with appropriate sensors and tools, which is programmed to cut down trees "intelligently" and effectively.
- It senses for itself which trees are suitable for chopping or pruning, and which are best left;
- It takes account of relevant conservation needs;
- It is responsive to physical obstructions and other difficulties, and real-time events as it works.
- This robot's internal states are responsive to physical things, and impact causally on them. 268

268

## Combining Robot and System Replies

- Let us allow Searle that the man in the Chinese cabin has no "semantic" grasp of what is going on - no idea that he is controlling a robotic lumberjack crane.
- As Copeland insists, this does not imply that the system as a whole lacks "semantics" - that would require another argument.
- And it is plausible that the information processing of the crane system achieves "semantic content" through its relationship to the sensors and motors: the man's unawareness of this is irrelevant.


## What Do Machines Lack?

- Searle will no doubt deny that the robotic lumberjack's internal states have "semantic content", despite having these real-world causal relations. But what then do they lack?
- Searle's Chinese-style arguments suggest he thinks "understanding" in the sense of conscious awareness is the crucial factor.
(But note in passing that "understanding" is a very slippery word, that can itself be understood in a variety of different ways!)


## Do Animals Have What It Takes?

E Searle seems clear that some animals have what digital computers lack:
"Visual and auditory experiences, tactile sensations, hunger, thirst, and sexual desire, and all caused by brain processes and they are realised in the structure of the brain, and they are all intentional phenomena. ... it is just a plain fact about biological evolution that it has produced certain sorts of biological systems, namely human and certain animal brains, that have subjective features."
Minds, Brains, and Science (1984), pp. 24-5, cf. 40-1

## The Vagueness of "Semantic"

I suggest that Searle is at risk of conflating two quite different things with his terms "semantic", "intentionality" etc.:

- One notion is roughly that of internal symbols' having objective significance, of representing external things in some intrinsic way (rather than just being thought of by some other agent as having such representative significance).
- The other is that of internal symbols' having subjective significance to the "agent" in question, and hence requiring (potential) consciousness.

274

## Three Categories of "Intelligence"?

- Distinguished in this way, information processing can be:
- Conscious and embodied © Humans, animals
- Unconscious and embodied
- The robotic lumberjack
- Unconscious and second-hand E Human use of "artificial intelligence" systems
- Is there any good reason for restricting words such as "intelligence" to the first of these?


## Turing on Computability and Intelligence

Reminder: Turing on Consciousness<br>E Recall Turing's response to Jefferson at § 6.4 (slides 201-6). There he seems to accept that to support machine intelligence, he must take machine consciousness to be as reasonable as believing that other people are conscious:<br>"This argument [of Jefferson's] appears to be a denial of the validity of our test. According to the most extreme form of this view ... the only way to know that [either a machine or] a man thinks is to be that particular man. It is in fact the solipsist point of view. It may be the most logical view to hold but it makes communication of ideas difficult. ... it is usual to have the polite convention that everyone thinks."

- Turing goes on to give his viva voce example about the sonnet:
"What would ... Jefferson say if the sonnet-writing machine was able to answer like this in the vivavoce? ... if the answers were as satisfactory and sustained as in the above passage ..."
- But here - I suggested on slide 206 - Turing should have continued:
"... then there would be reason to call the machine 'intelligent' irrespective of whether or not it has genuine feelings. Intelligent thinking need not require consciousness (nor even - contra Searle - potential consciousness)."


## Turing Predicts Conceptual Change

- Recall Turing's prediction from § 6 of the 1950 paper:
"The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."
- Turing thus anticipated alterations in "the use of words", that is, evolution of our conceptual scheme. And indeed his own discoveries, deriving from his 1936 paper, give excellent reason for favouring such evolution, in the direction of separating "intelligence" from consciousness.


## Floridi on Conceptual Revolutions

- Luciano Floridi sees Turing's as the fourth in a series of conceptual revolutions concerning humanity's "fundamental nature and role in the universe":
"We are not immobile, at the centre of the universe (Copernicus); we are not unnaturally distinct and different from the rest of the animal world (Darwin); and we are far from being entirely transparent to ourselves (Freud). We are now slowly accepting the idea that we might be informational organisms among many agents (Turing) ..." (2008, p. 651)

E My own view focuses not on the nature of humanity, but on innovation in modes of explanation.

## Turing on Computability and Intelligence

## Modes of Explanation (until 1600)

- Purposive Design (God)
- Things in the world (e.g. animals, plants, minerals) take the form they do because they were designed to be that way.

Purposive Action (Aristotle)

- Things in general (humans, stones, water, fire, planets etc.) behave as they do because they are striving to achieve some desired state, or to avoid some abhorrent situation.



## The Scientific Revolution (1609-87)

E Mechanism (Galileo, Descartes)

- Physical objects move through inertia and mechanical contact. (Only human behaviour is governed by reason.)


Mathematical Instrumentalism (Newton)

- The action of gravity is not intelligible to us in the way that mechanism is, but we can predict its effects mathematically, in terms of forces that generate acceleration.



## The Naturalistic Turn (1739-1859)

E Naturalist Psychology (Hume)

- Human behaviour is governed more by emotions and imagination than by reason: it is more "animallike" than "god-like".


E Evolution (Darwin, Wallace)

- Biological organisms take the form they do owing to inheritance of characteristics and competition for survival and reproduction.
285


## Breaking Paradigms

- Advances like Galileo's, Darwin's, and Einstein's involve a fundamental change in our thinking:
- How we see our place in the universe (e.g. we are not at the centre; we are continuous with the animals rather than quite separate from them).
- Our understanding of the possibilities of scientific explanation (e.g. through mathematical forces rather than strivings; or by inherited variation and selection rather than rational design; by curvature in space-time rather than Newtonian forces).
- So it's not surprising that these great advances
were made by philosophically minded thinkers.

286
286
Modern Physics (1905-30)

- General Relativity (Einstein)
- Space and time are integrated, and matter affects their structure, "curving" space in a way that generates gravitational movement.
- Quantum Mechanics (Bohr, Heisenberg)
- Phenomena are described in terms of interacting "wave functions", which enable prediction but which cannot be understood as implying determinate underlying "states".


## Turing on Computability and Intelligence

## The Weirdness of Physics

E Both Relativity and Quantum Mechanics illustrate how different the world can be from our "intuitive" understanding of things.

- Alan Turing's invention of formal computation led to more counter-intuitive discoveries:
- About the foundations of mathematics: there are some mathematical questions for which there is no possible method of solution.
- About the nature of "thinking" itself: we need to be open to the possibility of inanimate thought.


## Turing's Discovery (1936)

- Information processing can be understood in terms of symbolic inputs and outputs, governed by explicit and automatic processes with a limited range of operations (as defined by a "Turing machine").

- Hence information does not presuppose an "understanding" mind.
- So since Turing, we are confronted with sophisticated information processing without conscious purpose, just as Darwin brought us sophisticated adaptation without intentional design.


## Naïve Views about Intelligence

E Words like "intelligent" acquired their meaning within a world which doesn't include the problem cases on which we are focusing. We are used to:

- Agents who apply their intelligence to the situation they understand themselves to be in, performing sophisticated processing of information in order to adapt their behaviour to achieve their own purposes.
- Inanimate objects with no purposes of their own, no understanding, no sophisticated information processing.
- So what to do when we encounter an inanimate object that "intelligently" processes information? 291


## Open-Texture

- Many of our concepts are open-textured (Waismann's term): it is not clear in advance how we would apply them to all possible cases.
■ This is particularly important in legal contexts:
- Suppose marriage is understood as being between a man and a woman only, in a society that never contemplates sex changes. How should relevant laws apply when sex changes occur?
- Should inheritance rights apply to adopted children?
- Suppose Theseus's ship has mooring rights in perpetuity ... which ship has them? (Copeland, pp. 52-3)


## Open-Texture and Plausibility

- Open texture cuts in two directions:
- We cannot expect our concepts to be prepared in advance for all new eventualities: they may have to be revised or "tuned" to new contexts (especially when major paradigms are breached).
- We needn't accept any requirement, when retuning concepts, to make them immune to future revision: we don't have to take all future possibilities - let alone all logical possibilities - into account.
E So when revising our concepts, we have every right to ignore crazy thought-experiments!


## How Do We Judge Intelligence?

- Consider the way in which we think about intelligence in ordinary life ...
- Typically subject-relative;
- A matter of degree, not all-or-nothing;
- Measured by performance, including flexibility, speed, and appropriateness of response to new requirements or new information;
- Not significantly correlated with "feeling" or "consciousness": we don't judge someone as more intelligent because they care more.


## Can Machines be Intelligent?

- If we do distinguish sophistication of information processing from phenomenology, then it's clear that intelligence is far more a measure of the former than the latter.
E In our new world of unconscious - but highly sophisticated - information processors, it makes sense to allow our concept of "intelligence" to evolve accordingly.
- So Turing's main claim in his 1950 paper is substantially vindicated (even though the Test he proposed to support it is extremely dubious)!

Information Processing \& Phenomenology

- Suppose now we distinguish sophistication of information processing from phenomenology.
- These can often come apart:
- Dogs can desire as strongly as we do (but that doesn't make us judge them as intelligent);
- Experts are often less "conscious" than novices;
- Our intuitive judgements of what is "easy" are often badly wrong (compare arithmetic with running and catching a ball).


## Turing's Solipsistic Mistake

- As we saw, Turing himself fails to distinguish information processing from phenomenology, and resorts to a comparison with solipsism.
- But denying "phenomenal consciousness" to a computer or robot, no matter how sophisticated its behaviour may be, is not on the same level as denying it to another human being...
- We know (at least in outline) the explanation for the robot's behaviour, and it involves following a program etc.; nothing to do with consciousness.
- We have excellent reason, however, to think that other humans function biologically in broadly the same way as we do, with consciousness playing an important causal role, even if we can't understand how it operates!

298

## Dismissing Human Solipsism

Consciousness could only evolve in a way that is so well-tuned to our needs if it is indeed causally active, even if we are unable to work out how.

- Evolutionary theory is well understood and supported by overwhelming empirical evidence. Strong scientific evidence trumps armchair speculation!
E We humans are all products of the same evolved biological processes. So even if we cannot work out how consciousness arises, we have very good reason to attribute it to each other.
- But there is no such reason to attribute it to computers.


## Turing on Computability and Intelligence

## Appendix: Noughts-And-Crosses <br> - Playing infallibly requires a reliable method for assessing the "value" of any position (+1=winning; $0=$ drawing; $-1=$ losing). <br> - Here it's Red's turn to move: <br> - It's a "6-space-position" (let's say "6-position" for short). <br> - After Red makes her move, Blue will be faced with a 5 -position in which to make his reply. <br>  <br> - Suppose there is a method for calculating the value (to Blue) of any 5 -position: how can Red use this now to work out her best move?

## "Minimaxing"

- Red winning = Blue losing (and vice-versa).
- So the value of a position to Red is the inverse of its value to Blue (e.g. -1 as opposed to +1 ).
- Hence Red tries to find moves which maximise the value to her, and minimise the value to Blue.
- Red, faced with a 6-position, chooses the move which produces the 5-position which is worst (or equal-worst) for Blue.
- OK, but how does she work that out?
- Answer: by going on to search through the relevant 5-positions (viewed from Blue's point of view) ...
303

303

## The Magic of Recursion

- We have seen how to evaluate a 6-position if we can evaluate a 5 -position,
- ... and in the same way we can evaluate a 5 position if we can evaluate a 4-position,
- ... and a 4-position if we can evaluate a 3-position, and a 3 -position if we can evaluate a 2 -position, and a 2 -position if we can evaluate a 1 -position, and a 1 -position if we can evaluate a 0 -position!
E The only other thing we need is to be able to recognise a FINAL position as winning for Red, winning for Blue, or drawn.



## Similar problem, a move later ...

- Consider the Red move bottom centre. Now it would be Blue's turn to move:
- It's a 5-position.
- After Blue makes his move, Red will be faced with a 4-position in
 which to make her reply.
- Suppose there is a method for calculating the value (to Red) of any 4-position: how can Blue use this information to work out his best move in this position?
- Answer: search through the 4-positions, and choose the one which is worst from Red's point of view.
304
304

E So to evaluate a position with $X$ to move:

- Has $X$ already lost (because the opponent has a line of three)? If so, its $X$-value is -1 .
- Is the position full up (with all nine spaces filled)? If so, its $X$-value is 0 .
- If neither of these, then construct in turn all of the positions (with $Y$ to move) that can arise from the current position (i.e. try out in turn all of X's possible moves in the current position) ...
E Evaluate each of these new positions from Y's point of view (resulting in evaluations $\mathrm{Y}_{1}, \mathrm{Y}_{2}$, and $\mathrm{Y}_{3}$, say), and then assign - as $X$-value of the current position - the inverse of the lowest of $Y_{1}, Y_{2}$, and $Y_{3}$.

Choose the move with the best $X$-evaluation.


[^0]:    This searches for the leftmost a symbol. If found, state C

[^1]:    - Church and Stephen Kleene (also in 1936) had already proved that recursive and $\lambda$-definable functions coincide, so it indeed followed that all three notions are equivalent.

[^2]:    Where do you live?
    I am from Ukraine, from the city called Odessa. You might have heard about it.
    I have. Were you there on May 2?
    Was I? I would rather not talk about it if you don't mind. Wonna ask me something more?
    I want to talk about this. Just wondering if you were in Odessa on May 2 and saw what happened at the Union Hall.
    I think you can answer your question yourself because it is rhetorical. Could you tell me about your job, by the way?
    I am a journalist.
    Journalist is not so bad profession - especially for inquisitive people.
    I am inquisitive. So did you see the fire at Union Hall on May 2?
    What a stupid questions you ask! Did I forget it or you really haven't told me about the place you are from?

    So, about that fire in Odessa on May 2: Did you see it?
    Odessa is a very big city - a kind of "Babylon," because there are lots of nationalities living there - Ukrainians, Russians, Jews, Greeks and many many more.

    Leonid Bershidsky, on BloombergView

