Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Computer Science and Philosophy, TT 2025



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

1

Role of the Course

- This course is part of the "Introduction to Philosophy" for Prelims in *Computer Science & Philosophy.*
  - The 3-hour examination will contain six (possibly "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

2

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.

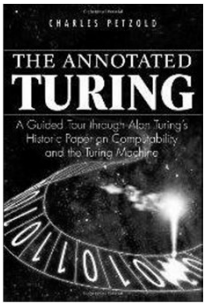
The lectures will be given in the order 1, 3, 2, 4, 5, 6, 7, 8 so as to space out your study of the 1936 paper (lectures 3 to 5).

3

3

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

Recommended Reading on Lectures 1 to 5

- Andrew Hodges, *Alan Turing: The Enigma of Intelligence* (Hutchinson 1983, 586 pp.) An excellent, influential biography.
- Jack Copeland (ed.) *The Essential Turing* (OUP 2004, 613 pp.) Authoritative collection of Turing's main papers, with commentary.
- S. Barry Cooper and Jan van Leeuwen (eds), *Alan Turing: His Work and Impact* (Elsevier 2013, 914 pp.) A rich collection of Turing's work with varied commentary from philosophers, mathematicians, computer scientists etc.
- Ernest Nagel and James R. Newman, *Gödel's Proof* (RKP 1959, 118 pp.; revised NYUP 2001, 129 pp.) Short and very approachable presentation of Gödel's theorem – will be referred to as "*NN*" in Lecture 2.
- Peter Smith, *Gödel Without (Too Many) Tears* (Logic Matters online 2022, 146 pp.) Goes far more deeply into Gödelian topics: difficult but very clear – will be referred to as "*GWT*" in Lecture 2.

5

5

Alan Turing

- 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

6

Turing on Computability and Intelligence

- (Later in his life, despite his heroic – but largely unknown – decryption of Nazi *Enigma* codes during the War, Turing was prosecuted for homo-sexuality, subsequently killing himself in 1954.)
- Turing reapplied to Cambridge in 1930, and was accepted for King's College, going up in 1931.
- 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

7

David Hilbert (1862-1943)



- Hilbert (born Königsberg) was perhaps the most influential mathematician of the early 20th century.
- He developed the *formalist* approach to the philosophy of mathematics, which sees mathematics as involving manipulation of symbols according to an agreed set of *formal rules*, and starting from given *axioms*.
- At the 1900 Paris International Congress of Mathematicians, Hilbert famously listed the most prominent unsolved problems of mathematics.

8

8

- Hilbert's list later grew to 23, several of which are discussed by Petzold:

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) …

   [$\aleph_0$ is the cardinality of the natural numbers $\mathbb{N}$, and $2^{\aleph_0}$ is the cardinality of the power set of $\mathbb{N}$ (i.e. the set of subsets of $\mathbb{N}$). But every subset of $\mathbb{N}$ corresponds to a unique sequence of binary digits – put "1" in the n^{th} place if n is in the subset, and "0" otherwise – and hence to a real number between 0 and 1 (which therefore has the cardinality of the continuum). Is this the smallest cardinal greater than $\aleph_0$? That claim – the *continuum hypothesis* – has since been shown to be neither provable nor disprovable within standard set theory.]

9

9

2. **Prove the consistency of arithmetic axioms**
   See Petzold p. 46, and discussion below.

8. **Riemann hypothesis and other problems including Goldbach's Conjecture**
   Goldbach is mentioned by Petzold on p. 47.

10. **Determination of the Solvability of a Diophantine Equation**
    These are polynomial equations *that must have integer solutions* (including "Fermat's last theorem" examples: $x^n + y^n = z^n$). For interesting discussion of this problem and the history of its (dis-) solution, see Petzold, chapters 1 and 18.

10

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": to devise an effective procedure which would demonstrate – in a finite time – whether any given mathematical sentence is, or is not, provable from a given set of axioms. "*The decision problem must be called the main problem of mathematical logic*" (p. 77).

11

11

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 …

12

12

Turing on Computability and Intelligence

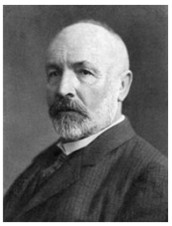
Various Types of Number

- All of the following sets of numbers are standardly used in mathematics:
  - Natural numbers [0?], 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.

13

13

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" …

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), referring to Hume's *Treatise* 1.3.1.5

15

15

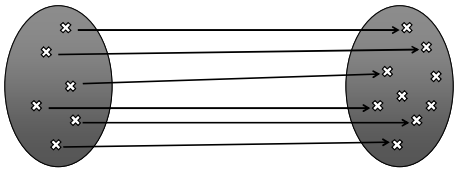
One-to-one Correspondence

- 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).

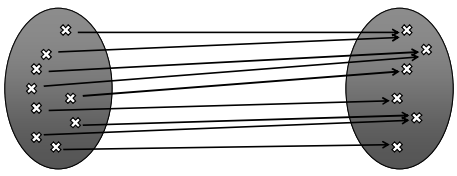
16

16

An Injective Function – no two arrows "hit" the same target



A Surjective Function – every element of the codomain is "hit"



17

17

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: A → 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

  $(|A| \geq |B|) \wedge (|B| \geq |A|) \;\therefore\; |A| = |B|$

18

18

Turing on Computability and Intelligence

Enumerability / Countability

- 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).
- In conformity with the obvious meaning of the term, we here take the "natural numbers" to start from 1 (though it's often more convenient to include 0, as we'll see in Lecture 2).
- Georg Cantor proved that the set of rational numbers $\mathbb{Q}$ and the set of algebraic numbers are both enumerable.

19

19

Rational Numbers

- A rational number is a number that can be expressed as a fraction of integers.
- 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… or 0.666… or 0.367367367…
- To convert to a fraction, sum the series:

$0.367367367\ldots = \frac{367}{1000} + \frac{367}{1000000} + \frac{367}{1000000000} + \cdots$

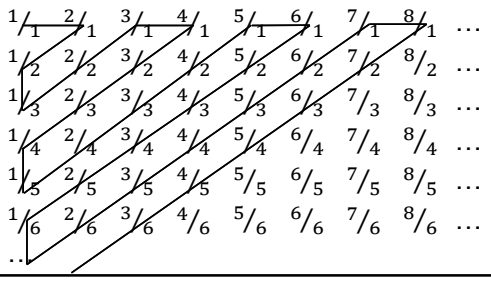
$= \frac{367}{\cancel{1000}} \times \frac{\cancel{1000}}{999}$ [using the formula "$a/(1-r)$"]

20

20

Rational Numbers are Enumerable

- Set out the fractions of positive integers in a grid, and define a surjection F: $\mathbb{N} \to \mathbb{Q}$ following the line: $F(1) = {}^1/_1$, $F(2) = {}^2/_1$, $F(3) = {}^1/_2$, etc.



21

21

- We thus list all of the fractions in the array, starting ${}^1/_1$, ${}^2/_1$, ${}^1/_2$, ${}^1/_3$, ${}^2/_2$, ${}^3/_1$, ${}^4/_1$ …
- So our surjection maps 1 to the 1st of these, 2 to the 2nd, 3 to the 3rd, 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).

22

22

√2 is Not Rational (Euclid)

- 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 $2n^2 = m^2$, and m^2 is even.

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

23

23

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 ${}^a/_b$ (with a & b integers) is the solution to the equation $bx = a$.
- Any *square root* or n^{th} *root* of a rational number is algebraic, being a solution to some equation of the form $bx^n = a$.

24

24

Turing on Computability and Intelligence

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:
  $n + |a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|$
- 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}, …, finally by a_0.

25

25

- 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:
  $-2x + 0 = 0, \quad -x - 1 = 0, \quad -x + 1 = 0,$
  $x - 1 = 0, \quad x + 1 = 0, \quad 2x + 0 = 0,$
  $-x^2 + 0 = 0, \quad x^2 + 0 = 0$

  Rank 4:
  $-3x + 0 = 0, \quad -2x - 1 = 0, \quad -2x + 1 = 0,$
  …

26

26

- 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 1st in the list, 2 to the 2nd, 3 to the 3rd, 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.

27

27

Transcendental Numbers

- A *transcendental number* is a number which is not algebraic – so it is not a solution to any algebraic equation.
- π and e are both transcendental, as are $\sin(\alpha)$, $\cos(\alpha)$, and $\tan(\alpha)$, for any non-zero algebraic value of α.*
- Likewise $\ln(a)$ and a^b if a is algebraic (and not 0 or 1), and b is irrational algebraic.

* Note that α represents an angle *in radians, not degrees*.

28

28

Supposing that Real Numbers are Enumerable …

- 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: $0.a_{11}a_{12}a_{13}a_{14}a_{15}a_{16}a_{17}a_{18}$ …
a_2: $0.a_{21}a_{22}a_{23}a_{24}a_{25}a_{26}a_{27}a_{28}$ …
a_3: $0.a_{31}a_{32}a_{33}a_{34}a_{35}a_{36}a_{37}a_{38}$ …
a_4: $0.a_{41}a_{42}a_{43}a_{44}a_{45}a_{46}a_{47}a_{48}$ …
a_5: $0.a_{51}a_{52}a_{53}a_{54}a_{55}a_{56}a_{57}a_{58}$ …
… …

29

29

… Leads to Diagonal Contradiction

- Consider now the "diagonal" number d that we get if we take the digits in order down the list's long diagonal ($a_{11}a_{22}a_{33}a_{44}a_{55}\ldots$), but ***tweaking*** them so always $d_n \neq a_{nn}$, e.g. using this simple rule:

  If a_{nn}=0, then d_n=1, otherwise d_n=0.
- Clearly the number d will differ from the n^{th} number in the list at the n^{th} decimal place, for all n. Hence d is a real number between 0 and 1 that does not occur in the list of all such numbers … CONTRADICTION!!

30

30

Turing on Computability and Intelligence

- 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 <u>all</u> sets, deriving a contradiction from the assumption of a surjective function from *any* set to its power set.

31

31

- Suppose F is a surjection from *A* to $\wp(A)$, 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:

	A[1]	*A*[2]	*A*[3]	*A*[4]	…
$F(A[1])$	✓	×	×	×	…
$F(A[2])$	×	✓	×	×	…
$F(A[3])$	✓	✓	×	×	…
$F(A[4])$	×	×	✓	×	…

Elements of *A* that are members of the corresponding subset have a <u>tick</u> on the long diagonal. But if we take the set of elements that have <u>crosses</u> down the long diagonal, we get a contradiction if we suppose that this set is anywhere in the list. (For if it has a cross on the long diagonal, it should have a tick, and vice-versa.)

32

32

An Infinity of Infinities

- The proof of Cantor's Theorem can work equally well without the explicit diagonalisation, if we simply consider the set:

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

- Suppose that $S = F(n)$ for some $n \in A$. Then *n* satisfies the criterion for membership of $F(n)$ if and only if it is not a member of $F(n)$ – contradiction!
- Since, in general there cannot be a surjection from *A* to $\wp(A)$, 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).

33

33

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.

34

34

- 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."

35

35

Jules Richard's Paradox about Definable Numbers

- Another Cantorian "diagonal" paradox was devised by the French mathematician Jules Richard in 1905, and later discussed by logicians including Gödel and Church.
- It may have provided a key inspiration for Turing's 1936 paper, as will be explained in subsequent lectures.



Jules Richard (1862-1956)

36

36

Turing on Computability and Intelligence

- Imagine creating (e.g. by means of a suitable computer program, maybe one that continues indefinitely) a list of *all meaningful English phrases that succeed in specifying a positive decimal number less than 1.0*. Here are a few examples of such phrases and their corresponding numbers:

"Point one recurring"	0.111…
"Point one six"	0.16
"Point two seven"	0.27
"Point three recurring"	0.333…
"Point four one"	0.41
"Half"	0.5
"Pi divided by six"	0.523598775…
"The reciprocal of root two"	0.707106781…
"Point eight"	0.8

37

37

- *English phrases are enumerable*. Imagine putting our number-specifying phrases into a single ordered list, starting with those that are just one word long, then two words, then three, etc., and for each length, putting them into *alphabetical order*. List the defined numbers accordingly, including their entire decimal expansions:

(1)	"Half"	0.500000000…
(2)	"Point eight"	0.800000000…
(3)	"Point four one"	0.410000000…
(3)	"Point one recurring"	0.111111111…
(3)	"Point one six"	0.160000000…
(3)	"Point three recurring"	0.333333333…
(3)	"Point two seven"	0.270000000…
(4)	"Pi divided by six"	0.523598775…
(5)	"The reciprocal of root two"	0.707106781…

38

38

- Then construct the diagonal number of the entire array, with Cantorian *tweaking* (i.e. 0 becomes 1, other digits become 0):

"Half"	0 . 5 0 0 0 0 0 0 0 0 …
"Point eight"	0 . 8 0 0 0 0 0 0 0 0 …
"Point four one"	0 . 4 1 0 0 0 0 0 0 0 …
"Point one recurring"	0 . 1 1 1 1 1 1 1 1 1 …
"Point one six"	0 . 1 6 0 0 0 0 0 0 0 …
"Point three recurring"	0 . 3 3 3 3 3 3 3 3 3 …
"Point two seven"	0 . 2 7 0 0 0 0 0 0 0 …
"Pi divided by six"	0 . 5 2 3 5 9 8 7 7 5 …
"The reciprocal of root two"	0 . 7 0 7 1 0 6 7 8 1 …
Diagonal digits are:	5 0 0 1 0 3 0 7 1 …
So diagonal number is:	0 . 0 1 1 0 1 0 1 0 0 …

39

39

The Paradoxical Result

- The diagonal number has been defined in such a way that it cannot be in the list.
- So if we imagine the list being constructed with *all meaningful English phrases that succeed in specifying a positive decimal number less than 1.0* (rather than just a handful of examples), it follows that the diagonal number which results cannot be defined by any English phrase.
- Yet in describing how the number is to be constructed, we have defined it in English! …

40

40

"Take all meaningful English phrases that succeed in specifying a positive decimal number less than 1.0. Put these into a single (potentially infinite) list, starting with those that are just one word long, then two words, then three words, etc., and for each length, put them into *alphabetical order*. List the defined numbers accordingly, including their entire decimal expansions. Then construct the diagonal number of the entire array, by starting with '0.' and then taking in turn the nth decimal digit of the nth number in the array, and appending '1' if that digit is '0',and appending '0' otherwise."

41

41

Two Plausible Responses

1. The notion of *a meaningful English phrase that succeeds in specifying a number* is not well-defined (as shown by its leading to paradox).
2. It is specifically problematic to attempt to define a number by reference to a collection of which it is itself a part (Russell called this the *Vicious Circle Principle*). In the following would-be list (L) of three numbers, for example, there is no consistent third digit for the third number:

(L): 0.012
0.345
The three-digit diagonal number of list (L)

42

42

Turing on Computability and Intelligence

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

43

Useful Resources on Gödel's Theorem

- As noted in Lecture 1, the following two books (and especially *GWT*) will be referenced here:
  - Ernest Nagel and James R. Newman, *Gödel's Proof* (RKP 1959, 118 pp.; revised NYUP 2001, 129 pp.)
    Short and very approachable presentation of Gödel's theorem – will be referred to as "*NN*" in Lecture 2.
  - Peter Smith, *Gödel Without (Too Many) Tears* (Logic Matters online 2022, 146 pp.) Goes far more deeply into Gödelian topics: difficult but very clear – will be referred to as "*GWT*" in Lecture 2.
- For an excellent overview of this topic area, see:
  - Peter Smith, *Beginning Mathematical Logic: A Study Guide* (Logic Matters online 2024, 184 pp.), Chapter 6: "Arithmetic, computability, and incompleteness".

44

44

Recall Hilbert's Ambitions
(from lecture 1, slide 11)



- 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": to devise an effective procedure which would demonstrate – in a finite time – whether any given mathematical sentence is, or is not, provable from a given set of axioms. "*The decision problem must be called the main problem of mathematical logic*" (p. 77).

45

45

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.

46

46

Key Semantic Notions

- A *theory* is *sound* iff its axioms are true (on the standard interpretation of its language) and its proof system is truth-preserving, so all its theorems are true. (*GWT*, p. 11).
- An *interpreted theory* is *complete* iff every sentence that is true in the relevant domain, and on the standard interpretation, is provable within the theory.
- A *proof-system* with derivability relation $\vdash$ is *sound* for a logical consequence relation $\models$ iff it is truth-preserving, so that for all Γ, φ, if $\Gamma \vdash \varphi$ then $\Gamma \models \varphi$.
- A *proof-system* with derivability relation $\vdash$ is (*semantically*) *complete* for a logical consequence relation $\models$ just in case, for all Γ, φ, if $\Gamma \models \varphi$ then $\Gamma \vdash \varphi$.

47

47

Syntactic Formalism

- In Gödel's time, the notion of truth in mathematics was considered problematic (*GWT* p. 12), so it was common to interpret the relevant notions *syntactically* (i.e. in terms of structural relationships between formulae) rather than *semantically* (i.e. in terms of truth and meaning).
- Thus understood,
  - a *consistent* system is one in which it is impossible to prove *both* a sentence P *and* its negation $\neg$P;
  - a *complete* system is one in which it is always possible *either* to prove P *or* to prove $\neg$P, for any sentence P that is expressible within the system. (This is more precisely called *negation completeness*, to distinguish it from *semantic completeness*)

48

48

Turing on Computability and Intelligence

Consistency and Completeness

- Thus *consistency* (we cannot prove both *P* and ¬*P*) and *negation completeness* (we can prove either *P* or ¬*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 negation complete and *sound* system of arithmetic (i.e. with *true* axioms and *truth-preserving* rules), then any arithmetical sentence would be provable if, and only if, it is true – such a system would be *semantically complete with respect to the standard model of arithmetic* (see *GWT* pp. 6-7 for more explanation of this notion). A major part of Hilbert's dream would thus be realised.

49

49

Gödel, Completeness and Incompleteness

- In 1930, a year before his famous Incompleteness Theorem, Gödel proved a Completeness Theorem!
  - This demonstrated that *first-order logic* is *semantically complete*, i.e. every formula of first-order logic which is logically valid (i.e. "true in all models") is formally provable using the standard rules of first-order logic.
- This is not in tension with his famous 1931 Proof:
  - This demonstrated that *any sound and recursively axiomatized theory able to express basic arithmetic* must be *negation incomplete* – i.e. for some formula G, neither G nor ¬G is provable in the system – and also *semantically incomplete with respect to the standard model of arithmetic* – because this unprovable formula will be true.

50

50

Formal Theories of Arithmetic

- The kind of system that Hilbert had in mind was that developed by Alfred North Whitehead and Bertrand Russell in 1910-1913, in their three-volume *Principia Mathematica*.
- Note also the title of Gödel's 1931 paper which presented his first incompleteness theorem: "On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems".
- However we will focus – as do most modern treatments – on a simpler system commonly known as "Peano Arithmetic" or "PA".

51

51

Peano Arithmetic ("PA")

- PA consists of a specific *language* (here "L_{PA}"), which we understand with its *standard interpretation* (referring to numbers and numeric operations etc.), and a *proof system* which can take various forms, but should conform to classical first-order logic.
- L_{PA} includes the language of first-order logic with identity (including connectives, quantifiers, variables, brackets, and '='), plus the special arithmetical symbols '0' (zero), 's' (the successor function), '+' and 'x'. Quantifiers ('∀' and '∃') are understood as ranging over the natural numbers (here including 0).
  - For the even simpler systems "Baby Arithmetic" (BA) and especially "Robinson Arithmetic" (Q), see *GWT* ch. 6.

52

52

Proofs in PA

- The *proof system* of PA can be any standard version of classical first-order logic with identity, including the familiar systems of natural deduction rules. But when theorising about PA, it is often most convenient to use instead only a minimal set of rules (e.g. *modus ponens* and *universal generalisation*) together with a set of *axioms* that express the desired logical relations (e.g. between structured formulae or identity statements), and also the relations of numbers.
- A valid proof in this kind of axiomatic system is typically understood as being just a sequence of formulae, each being either an axiom or an immediate implication of one or more preceding formulae. Then the proved formula is simply the last in the list.

53

53

Axioms of Peano Arithmetic

- Standard arithmetical axioms of PA, which can be introduced at any point in a proof, are:

1. $\forall x\ (0 \neq \mathrm{s}x)$
2. $\forall x \forall y\ (\mathrm{s}x = \mathrm{s}y \rightarrow x = y)$
3. $\forall x\ (x + 0 = x)$
4. $\forall x \forall y\ (x + \mathrm{s}y = \mathrm{s}(x + y))$
5. $\forall x\ (x \times 0 = 0)$
6. $\forall x \forall y\ (x \times \mathrm{s}y = (x \times y) + x)$
7. $\{\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(\mathrm{s}x))\} \rightarrow \forall x \varphi(x)$
   (where φ is an open well-formed formula that has "x" free – this is known as the induction schema)

These are to be distinguished from what are known as "The Peano Axioms", which we'll meet in Lecture 5, slide 215 (see also Petzold's book, p. 223).

54

54

Turing on Computability and Intelligence

A Simple Proof

- The numeral for 2 is "ss0", and the numeral for 4 is "ssss0". So the arithmetical statement "2+2=4" is expressed by the formula:

  ss0 + ss0 = ssss0

- A proof of this will be a sequence of formulae leading to it, where all of the formulae in the sequence are either axioms of PA or are validly derived from previous formulae in the sequence. In the current case this can be done using only substitution and dropping of brackets (and here we shortcut by omitting the axioms themselves and going straight to substitution instances):

55

Proving 2+2 = 4

1.	ss0 + ss0 = s(ss0 + s0)	From axiom 4
2.	ss0 + s0 = s(ss0 + 0)	From axiom 4
3.	ss0 + 0 = ss0	From axiom 3
4.	ss0 + s0 = s(ss0)	Line 3 into 2
5.	ss0 + ss0 = s(s(ss0))	Line 4 into 1
6.	ss0 + ss0 = ssss0	(drop brackets)

Later we'll introduce a convenient shorthand for the numerals, allowing us to write "2" instead of "ss0" and "4" instead of "ssss0".

56

Introducing Recursive Functions

- PA may seem to be extremely simple, but in fact it can represent a wide range of functions.
- Notice that our Axioms 3 to 6 define addition and multiplication *recursively*, by specifying the result for some *base case* (here 0), and then specifying the result for higher-value arguments in terms of results for lower-value arguments:

3. $\forall x\ (x + 0 = x)$
4. $\forall x \forall y\ (x + \mathrm{s}y\ =\ \mathrm{s}(x + y))$
5. $\forall x\ (x \times 0 = 0)$
6. $\forall x \forall y\ (x \times \mathrm{s}y\ =\ (x \times y) + x)$

57

- Recursive definitions can also easily be given of the *factorial* and *exponential* functions:

$0! = 1$
$\forall x\ (\mathrm{s}x! = x! \times \mathrm{s}x)$

$\forall x\ (x^0 = 1)$
$\forall x \forall y\ (x^{\mathrm{s}y} = x^y \times x)$

- A natural question to ask is: *what range of functions are definable in this sort of way, starting from the following 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^{th} of them, e.g. $P_2^4(12, 34, 56, 67) = 34$
  - Function *composition* (i.e. applying functions in turn)

58

Primitive Recursive Functions

- To generalise *recursion* itself, allowing for any number of arguments, we write $\vec{x}$ to represent any array of variables, and then function f will be said to be defined from functions g and h by *primitive recursion* if:

$$f(\vec{x}, 0) = g(\vec{x})$$
$$f(\vec{x}, \mathrm{s}(y)) = h(\vec{x}, y, f(\vec{x}, y))$$

- The *primitive recursive functions* are those functions that can be defined as just described (by composition and primitive recursion from primitive operations).
  - An equivalent formulation is in terms of *functions that are computable using "for" loops* (with neither the limit nor the loop counting variable being modifiable within the loop).

59

General Recursion and the Ackermann-Péter Function

- Not all computable or *general recursive* (μ-*recursive*) functions are *primitive* recursive, since some can be computed only with the aid of *"repeat … until"* or *"while"* loops (to perform "open-ended" searches).
- The most famous, devised by Wilhelm Ackermann (1928) and refined by Rósza Péter (1935), grows faster than any primitive recursive function:

$$p(0, y) = s(y)$$
$$p(s(x), 0) = p(x, s(0))$$
$$p(s(x), s(y)) = p(x, p(s(x), y))$$

60

Turing on Computability and Intelligence

Primitive Recursive Functions in PA and Gödel's Proof

- All primitive recursive functions can be expressed within L_{PA} (by providing appropriate definitions). Moreover PA's rules then enable the value of such a function to be calculated for any given inputs.
- Gödel's proof works in part by showing that an important range of *meta-logical* functions that we'll meet soon (notably *Wff* and *Prf*) are indeed primitive recursive, implying that they can be expressed *within* L_{PA} and that their correct application is accordingly computable within PA.

61

61

The "Arithmeticisation of Syntax"

- Suppose that we are able to devise some method of encoding logical formulae (or sequences of formulae) as individual numbers:
  - Encoding must be *algorithmic*, so that any formula (or sequence) can be encoded into a single code number by an entirely mechanical process.
  - Decoding must also be *algorithmic*, with a mechanical method for translating a code number back into the original formula (or sequence).
- Then the *syntactic* relationships between formulae can be calculated in terms of *arithmetical* relationships between their code numbers.

62

62

Gödel Numbering

- Here we follow *GWT*'s encoding scheme (pp. 78-9). This is more straightforward than Gödel's own (to which *NN* is closer). Following tradition, we call the codes *Gödel numbers* (henceforth signified by '#'):
  - The individual symbols are assigned codes as follows:

$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\forall$	$\exists$	=	(	)	0	s	+	$\times$	x	y	z	...
1	3	5	7	9	11	13	15	17	19	21	23	25	27	2	4	6	

    (This enables us to use subsequent even numbers for as many variables as we might need, e.g. $x_1, x_2, x_3, \ldots$)
  - Sequences of symbols are encoded by raising the n^{th} prime number (2, 3, 5, …) to the power of the code of the n^{th} symbol, and multiplying these all together.

63

63

- Thus the formula "$\exists x\,(x = sy)$" is encoded like this:

$\exists$	x	(	x	=	s	y	)
13	2	17	2	15	23	4	19

$$2^{13} \times 3^2 \times 5^{17} \times 7^2 \times 11^{15} \times 13^{23} \times 17^4 \times 19^{19}$$

- Then a "super-Gödel number" for a *sequence of formulae* can be derived by raising 2 to the power of the Gödel number of the first formula, 3 to the power of the Gödel number of the second formula, and so on, then multiplying these all together.
- To describe these numbers as "astronomical" would be a serious understatement! But crucially, they all have a unique factorisation into prime numbers, and that is the key to decoding them.

64

64

Decoding Gödel Numbers

- First, note that no distinct formulae (or sequences of formulae) can have the same Gödel number.
- Moreover it is straightforward "in principle" to decode a Gödel number (but not in practice: they're huge!):
  - First, factorise the number, identifying the *exponent* of each prime number within it (e.g. how many 2s, 3s, 5s, 7s etc. have been multiplied together to generate the number).
  - In a formula, the exponent of each prime will be the Gödel number of the corresponding *symbol* in the sequence (so $2^4 \times 3^{15} \times 5^6$ signifies "$y = z$", with 4 = #'y'; 15 = #'='; 6 = #'z').
  - In a sequence of formulae, the exponent of each prime will be the Gödel number of the corresponding *formula* in the sequence; then decode each formula as above.

65

65

A Simple Example: Substitution

- Let us now illustrate how a simple syntactic relationship between formulae can be represented arithmetically. We saw earlier how to derive the Gödel number of the formula:

  $\exists x\,(x = sy)$

  $2^{13} \times 3^2 \times 5^{17} \times 7^2 \times 11^{15} \times 13^{23} \times 17^4 \times 19^{19}$
- Here y is a free variable, and we can imagine substituting a particular numeral (say "ss0", representing 2) for y within the formula, yielding:

  $\exists x\,(x = sss0)$
- The Gödel number of this new formula is:

  $2^{13} \times 3^2 \times 5^{17} \times 7^2 \times 11^{15} \times 13^{23} \times 17^{23} \times 19^{23} \times 23^{21} \times 29^{19}$

66

66

Turing on Computability and Intelligence

- So here we have taken formula number:
  $2^{13}\times3^{2}\times5^{17}\times7^{2}\times11^{15}\times13^{23}\times17^{4}\times19^{19}$
  "$\exists x\ (x = sy)$"
  and within it we have substituted the variable with number:
  4 "y"
  by the L_{PA} numeral for the number:
  2 "ss0"
- This substitution has yielded formula number:
  $2^{13}\times3^{2}\times5^{17}\times7^{2}\times11^{15}\times13^{23}\times17^{23}\times19^{23}\times23^{21}\times29^{19}$
  "$\exists x\ (x = sss0)$"

67

67

- There will be a particular arithmetical function relating these four numbers. We can write:
  $2^{13}\times3^{2}\times5^{17}\times7^{2}\times11^{15}\times13^{23}\times17^{23}\times19^{23}\times23^{21}\times29^{19}$
  $= Sub(2^{13}\times3^{2}\times5^{17}\times7^{2}\times11^{15}\times13^{23}\times17^{4}\times19^{19}, 4, 2)$
- 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. 4 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.

68

68

Expressing *Sub* within PA

- *Sub* is a fairly complicated function, but it is provably primitive recursive, because its computation does not require any open-ended searches – the only loops needed would be "for" loops bounded by the length of the relevant formulae.
- Hence the rules for computing *Sub* could themselves be represented within L_{PA}. Let's use "Sub" as shorthand for the very long PA formula that would correspond to that arithmetical relationship (and note that the full formula for any instance of this would contain L_{PA} numerals for each of the four numbers concerned – e.g. '0' preceded by 's' repeated $2^{13}\times3^{2}\times5^{17}\times7^{2}\times11^{15}\times13^{23}\times17^{4}\times19^{19}$ times!).

69

69

Arithmetising Logical Properties

- In a similar way, after the Gödel number of any L_{PA} expression has been decoded, a program could clearly be devised to work out whether or not that sequence of symbols constitutes a "well formed formula" *(wff)* of L_{PA}.
- Moreover such a program need not involve any open-ended searches: any loops (e.g. to check for matching brackets) could be bounded by the formula length, meaning that the algorithm is *primitive recursive*.
- Hence, again, the algorithm could itself be represented in L_{PA}, and calculated using the rules of PA. Thus a predicate "Wff(x)" could be defined in PA, such that "Wff($\underline{n}$)" is true (or evaluates to 1 rather than 0 – see *GWT* pp. 3, 67) iff n is the Gödel number of a *wff* of L_{PA}.

70

70

- Likewise, after the super-Gödel number m of a sequence of L_{PA} formulae is decoded, a program could be devised to check systematically whether:
  - All of the encoded formulae are *wffs* of L_{PA};
  - Each such *wff* is either an axiom of PA, or follows from one or more previous *wffs* by the rules of PA;
  - The final *wff* has some specific Gödel number n.
- Again this would not require any open-ended searches, and is thus *primitive recursive*. So this algorithm could be represented in L_{PA}, and calculated using PA's rules. Thus a predicate "Prf(x,y)" could be defined in PA, such that "Prf($\underline{m}$,$\underline{n}$)" is true iff m is the super-Gödel number of a proof of the *wff* with Gödel number n.
  - For more detailed explanation of all this, see *GWT* pp. 80-81.

71

71

A Note on Numbers and Numerals

- We've just been talking a bit loosely about predicates such as *Wff(n)* and *Prf(m,n)* being defined in PA, where n is a Gödel number of some formula and m is a super-Gödel number of some sequence of formulae.
- But for these predicates to be defined in L_{PA}, they need to operate on the *numerals* $\underline{m}$ and $\underline{n}$ (where, as we saw, the L_{PA} numeral for 4 – shorthand $\underline{4}$ – is "ssss0").
- To distinguish, we shall denote the corresponding L_{PA} predicates as "Wff($\underline{n}$)" and "Prf($\underline{m}$,$\underline{n}$)" (upright, not italic).
  - Thus formula φ is a well-formed formula of L_{PA} iff, within PA, Wff(#φ) evaluates to 1 (i.e. true).
  - For similar moves in *GWT*, see pp. 18-19, 82-83.

72

72

Turing on Computability and Intelligence

Gödel's Theorem (1931)



- Kurt Gödel's First Incompleteness Theorem:
  - Informally: In any sound axiomatic theory sufficiently rich to enable the expression and proof of basic arithmetic sentences, it will be possible to construct an arithmetical sentence *G* such that neither *G*, nor its negation, is provable in that theory.
  - More Rigorously (see *GWT* pp. 12, 89): Suppose *T* is a formal theory (like PA) whose language contains the language of basic arithmetic and which is axiomatized primitively recursively. Then if *T* is sound (i.e. true axioms and valid inference rules), there will be a true sentence G_T of basic arithmetic such that $T \nvdash G_T$ and $T \nvdash \neg G_T$, so *T* is *negation incomplete*.

73

73

The Proof Strategy

- We have seen how formulae of PA, and sequences of formulae of PA, can be encoded as single "Gödel numbers" (this can be done in many different ways, as long as coding and decoding are primitive recursive).
- We have also seen, in outline, how a predicate Prf(*m*,*n*) could be defined in PA, to test whether *m* is the super-Gödel number of a proof of the *wff* with Gödel number *n*.
- We now aim to devise a mathematical formula *G* which is true if, and only if, there is no sequence *S* which yields a valid PA proof of *G* itself. This will be something like "$\neg\exists x$ Prf(x, #*G*)", where #*G* is the *numeral* for *G*'s own Gödel number. It says, in effect, "There is no x such that x is the super-Gödel number of a sequence that proves *G*."

74

74

Diagonalisation

- To derive our desired formula *G*, it will help to employ a construction called *diagonalisation*.
  - This echoes Cantor's theorem, where we asked whether α is an element of the set *F*(α); for more, see *GWT* pp. 26, 83.
- Suppose, then, that we have a wff "φ(x)" of L_{PA} in which the variable x is *free* (i.e. not bound by a quantifier). Then the *diagonalisation* of "φ(x)" is the wff "φ(#φ(x))" – that is, the same formula, but with the free variable substituted by the L_{PA} numeral for the Gödel number of "φ(x)" itself.
- The relation between the Gödel number of "φ(x)" and the Gödel number of "φ(#φ(x))" is primitive recursive, and so could be expressed in PA. We thus define "Diag(*m*,*n*)", which is true (i.e. evaluates to 1) iff *m* and *n* are thus related.

75

75

Adapting the Proof Predicate

- We now adapt our Prf(*m*,*n*) predicate, which evaluates to 1 iff *m* is the super-Gödel number of a PA proof of the formula with Gödel number *n* (and 0 otherwise).
- Our adapted predicate Prfd(*m*,*n*) will evaluate to 1 iff *m* is the super-Gödel number of a PA proof of *the diagonalisation of* the formula with Gödel number *n* (and 0 otherwise):

  $\text{Prfd}(m,n) =_{\text{def}} \exists z(\text{Prf}(m,z) \wedge \text{Diag}(n,z))$

  - Clearly *Prfd(x,y)* is a primitive recursive arithmetical relation; hence the expression of Prfd(*m*,*n*) within PA will indeed be possible (for more on this, see *GWT* p. 85).

76

76

Deriving the Unprovable Formula

- The following arithmetical formula in PA evaluates to 1 (i.e. true) if, and only if, *m* is *not* the super-Gödel number of a PA proof of *the diagonalisation of* the formula with Gödel number *n*:

  $\neg\text{Prfd}(m,n)$

- Correspondingly, the following open wff of PA – which we abbreviate to "U(*y*)" – evaluates to 1 if, and only if, *no number is the super-Gödel number of a PA proof of the diagonalisation of the formula with Gödel number y*:

  $\text{U}(y) =_{\text{def}} \neg\exists x(\text{Prfd}(x,y))$

- This wff itself has a particular Gödel number #U(*y*), and we now substitute the numeral #U(*y*) for *y* within it …

77

77

A Piece of Gödelian Magic

- So we move from:

  $\text{U}(y) =_{\text{def}} \neg\exists x(\text{Prfd}(x,y))$

  to a formula that we shall call "G" (in Gödel's honour):

  $\text{G} =_{\text{def}} \text{U}(\#\text{U}(y)) = \neg\exists x(\text{Prfd}(x,\#\text{U}(y)))$

- The ingenious point here is that *G is itself the diagonalisation of U(y)*, because it is the formula that results from substituting, for the free variable within U(y), the numeral for the Gödel number of U(y).
- So G is the diagonalisation of the formula with Gödel number #U(*y*), and if we now look at what G expresses, we see that G evaluates to 1 iff no number is the super-Gödel number of a PA proof of G itself!

78

78

Turing on Computability and Intelligence

An Alternative Path

- Since the slide on "Diagonalisation", we have been following what is now the standard route to deriving a Gödel sentence G (as used in *GWT*, for example), and we could now skip to "The Denouement" below.
- Gödel himself, however, derived his unprovable sentence in a different way, without defining diagonalisation as a specific operation, using a clever application of formula substitution.
- The next few slides illustrate this sort of process, following the method used in *NN* (but adapted to the PA formulation and Gödel numbering system used so far, which has been taken from *GWT*).

79

79

- Recall our variable substitution function *Sub*. For example, if F_y is a formula containing y (Gödel number 4) and F_n is the corresponding formula with y substituted by the numeral for *n*, then we have:

  $\#F_n = Sub(\#F_y, 4, n)$

- Imagine now that the function $Sub(x, y, z)$ is spelled out in gory arithmetical detail in PA – we earlier called this L_{PA} function "Sub" – and that we write out this expression with y in place of x and z, and $\underline{4}$ in place of y. This yields a complicated (but provably primitive recursive) arithmetical expression for:

  $\text{Sub}(y, \underline{4}, y)$

  which will soon play a key role.

80

80

Towards Another Unprovable Formula

- Recall also our PA proof predicate Prf, and consider the formula:

  $\neg\, \text{Prf}(\underline{m}, \underline{n})$

- This corresponds to the statement that the sequence of formulae with Gödel number *m* is <u>not</u> a proof in PA of the formula with Gödel number *n*.
- Likewise this arithmetical formula:

  $\neg\, \exists x\, \text{Prf}(x, \text{Sub}(y, \underline{4}, y))$

  corresponds to the statement that there is no proof in PA of the formula with Gödel number $Sub(y, 4, y)$.

81

81

$$\neg\, \exists x\, \text{Prf}(x, \text{Sub}(y, \underline{4}, y))$$

- This arithmetical formula corresponds to the meta-mathematical statement that
  - … there exists no sequence of formulae that constitutes a proof in PA of the formula with Gödel number $Sub(y, 4, y)$.
  - … or in other words, that formula (whose precise identity will obviously depend on the value substituted for "y") is not provable in PA.
- Now consider the arithmetical formula above, and suppose it has the Gödel number *n*.

82

82

Gödel's Magical Move

- We have that:

  $\#[\, \neg\, \exists x\ \text{Prf}(x, \text{Sub}(y, \underline{4}, y))\,] = n$

- Now consider the formula:

  (*G*) $\neg\, \exists x\, \text{Prf}(x, \text{Sub}(\underline{n}, \underline{4}, \underline{n}))$

  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:

  $\#[\, \neg\, \exists x\, \text{Prf}(x, \text{Sub}(\underline{n}, \underline{4}, \underline{n}))\,] = Sub(n, 4, n)$

83

83

(*G*) $\neg\, \exists x\, \text{Prf}(x, \text{Sub}(\underline{n}, \underline{4}, \underline{n}))$

- This arithmetical formula is true if, and only if:
  - there exists no sequence of formulae that constitutes a proof in PA of the formula with Gödel number $Sub(n, 4, n)$.
  - … or in other words, that formula is not provable within PA.
- But this arithmetical formula *itself* has the Gödel number $Sub(n, 4, n)$!
- So *G* is true $\leftrightarrow$ *G* itself is unprovable in PA!

84

84

Turing on Computability and Intelligence

The Upshot: G is Unprovable but True!

- The arithmetical sentence *G* encodes the statement that *G* itself is unprovable within PA.
- So if *G* were false, then it would follow (from this encoding) that *G* was not unprovable in PA:
  - Hence *G* would have to be provable in PA.
  - And if *G* were provable within a sound axiomatic system of arithmetic (which we are taking PA to be), then *G* would have to be true.
- Hence if *G* were false, we would have a contradiction. So *G* must be true and, given what it encodes, must also be unprovable within PA.

85

85

To spell this out further, the encoding means that:

G is true $\leftrightarrow$ G is unprovable in PA

$\therefore$ $\neg G \rightarrow G$ is provable in PA

But if *G* is provable in PA, and PA provides a sound representation of arithmetic, then *G* must be true, thus giving us:

$\neg G \rightarrow G$

So *G* cannot be false, and hence must be true. Yet if it is true, it is unprovable in PA (because it encodes the statement that it is unprovable in PA). So PA, if it is a sound axiomatisation of arithmetic, cannot be complete, for *G* will then be a true statement of arithmetic that cannot be proven in PA.

86

86

Gödel's Second Theorem

- We have seen that if PA is sound (and hence consistent), then sentence G cannot be provable within PA. If PA is inconsistent, however, then *G* (like any other formula) would be provable in PA.
- Let us represent PA's consistency as its inability to prove "0=s0" ("$\perp$" for short). Then using "$\text{Prov}_{PA}(\#\varphi)$" to mean that φ is provable in PA, we have:

  $\neg\text{Prov}_{PA}(\#\perp) \leftrightarrow \neg\text{Prov}_{PA}(\#G)$
- This follows straightforwardly from Gödel's First Theorem, so we might reasonably expect it to be provable in PA (though this proves to be very tricky: see *GWT* pp. 130 ff. for details). The upshot is that if PA is consistent, then just as it cannot prove G, nor can it prove its own consistency.

87

87

Disaster for Hilbert



- Gödel's First Theorem devastated Hilbert's ambition to establish a complete axiomatic theory of arithmetic, by proving that any such axiomatic theory, if consistent, must be incomplete.
- Gödel's Second Theorem threatened another of Hilbert's key ambitions – formulated in the wake of the discovery of Russell's and related paradoxes – to devise an adequate mathematical theory which, if rigorously applied, would be provably consistent, as a sound basis for future development. The Theorem showed that any such proof would have to use methods that could not be formalised in PA itself.

88

88

The *Entscheidungsproblem* Remains

- Gödel left the *Entscheidungsproblem* – Hilbert's celebrated *decision problem* – as unfinished business:
  - Gödel had shown that any *consistent axiomatic system of arithmetic would leave some arithmetical truths unprovable* (i.e. such a system could not be *complete*).
  - But there might still be some *effectively computable* decision procedure which would infallibly, in a finite time, reveal whether any given sentence *P* is, or is not, provable.

89

89

But 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.
- So to find the limits of "effective computability",
  - we need to devise a way of encompassing all possible *mechanical* methods of inference …
  - … and this role can be performed by Alan Turing's invention which we now call the *Turing machine*.

90

90

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 3

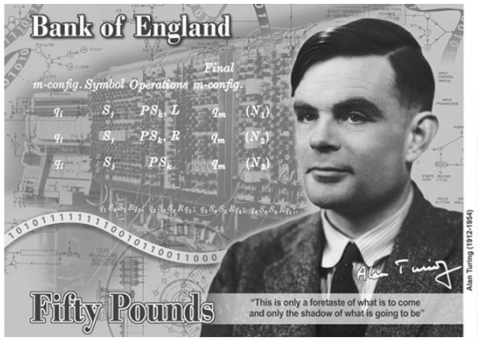
The Turing Machine, and the first four sections of the 1936 Paper



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

91

3.1 Introducing the Turing Machine



92

92

- The most important long-term contribution of Alan Turing's 1936 paper

  "On Computable Numbers, with an application to the *Entscheidungsproblem*"

  (Proceedings of the London Mathematical Society, 2nd 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.

References to the 1936 paper will be by page numbers in *The Essential Turing* (ed. Jack Copeland, OUP, 2004) and *The Annotated Turing* (Charles Petzold, Wiley, 2008)

93

93

§ 1 Computing Machines (*E* 59, *A* 68)

- Introducing the Turing machine
  - Most of (the short) § 1 is concerned with explaining the structure of a Turing machine. But first …
- 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 § 9, where he will argue in more detail why his "computing machines" can indeed capture all systematic methods available to a human "computer".

94

94

The Turing Machine as a Model of an Abstract Human Computer

- In § 9 of the 1936 paper, part I (*E* 75-7, *A* 190-4), Turing explains why his machine is a suitable model of human computation:

  "Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. … I assume then that the computation is carried out on … a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite …" (*E* 75, *A* 190-1)

95

95

- 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. …" (*E* 75-6, *A* 191)

96

96

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." (*E* 76, *A* 192)

97

97

"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."
(*E* 76-7, *A* 192-3)

98

98

"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, by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out." (*E* 77, *A* 193)

99

99

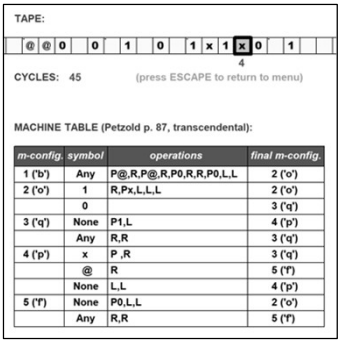
"We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an '*m*-configuration' of the machine. The machine scans *B* squares corresponding to the *B* squares observed by the computer. In any move the machine can change a symbol on a scanned square 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. The machines described do not differ very essentially from computing machines as described in § 2, and … a computing machine can be constructed to compute the same sequence …"
(*E* 77, *A* 194)

100

100

A Turing Machine in Action

Here we see a *Turtle System* Turing Machine simulator, 45 cycles through the processing of machine II from § 3 of his paper (*E* 62, *A* 87).



m-config.	symbol	operations	final m-config.
1 ('b')	Any	P@,R,P@,R,P0,R,R,P0,L,L	2 ('o')
2 ('o')	1	R,Px,L,L,L	2 ('o')
	0		3 ('q')
3 ('q')	None	P1,L	4 ('p')
	Any	R,R	3 ('q')
4 ('p')	x	P ,R	3 ('q')
	@	R	5 ('f')
	None	L,L	4 ('p')
5 ('f')	None	P0,L,L	2 ('o')
	Any	R,R	5 ('f')

It generates the binimal [binary fractional] number:
(0.) 0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 …
[which is roughly 0.179184 in decimal]

101

101

- The Turing machine consists of:
  - a long tape divided into squares, on which symbols can be printed and later erased,
  - a read/write head (as in a tape recorder) which can print (or erase) symbols and also read them.
  - a "state", taking any of a specified finite range of values – this is the machine's active memory. Computation starts from the first, *beginning*, state.
  - instructions in the form of a "machine table", which tell it what to do in each possible circumstance:
- On each cycle, depending on *the current state* and *the current symbol* (i.e. the symbol on *the current square*), the machine can:
  - Erase the symbol, or print a new one;
  - Move left or right on the tape;
  - Change to a new state.

102

Turing on Computability and Intelligence

Turing Machine Tables

- A ***strict*** machine table specifies one printing action and one move for each combination of current state and current symbol, yielding a table of "quintuples" (this is based on one of Petzold's examples, *A* 99):

State	Symbol	Print	Move	Next State
1 (b)	Blank	0	(N)	2
2 (i)	0	1	(N)	3
2 (i)	1	0	L	2
2 (i)	Blank	1	(N)	3
3 (r)	Blank	(Blank)	L	2
3 (r)	0	(0)	R	3
3 (r)	1	(1)	R	3

- But usually Turing abbreviates (e.g. allowing multiple actions in a single row, and omitting no-changes).

103

103

The "Configuration" (*E* 59, *A* 69-70)

- At any point, the machine can scan only one square on the tape (the "current" or "scanned" square).
- 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*"), and
  - the "*scanned symbol*"

  as the current "*configuration*" of the machine. The next action of the machine depends entirely on this.

104

104

The Repertoire of Actions (*E* 59-60, *A* 70-1)

- On each cycle, depending on this *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 in a single cycle (though this is restricted when the machines come to be regimented at § 5).

105

105

§ 2 Definitions (*E* 60, *A 72*)

- Turing focuses on *automatic* machines, whose behaviour is entirely determined by the configuration (though he also mentions *choice* machines, which are not fully determined).
- 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")

106

106

E/*F* Squares, and Marking (*E* 63, *A 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 β is on an *F*-square *S* and a symbol α is on the *E*-square next on the right of *S*, then *S* and β will be said to be *marked* with α. The process of printing this α will be called marking β (or *S*) with α" (end of § 3).

107

107

The Computed Number (*E* 60, *A 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*".
- "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.

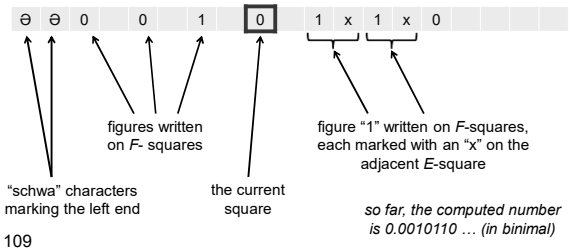
108

108

Turing on Computability and Intelligence

An Example of a Marked Tape (*A* 89)

- The tape is like this at one stage in the operation of Turing's machine "II" (table at *E* 62 / *A* 87):



109

109

"Circular" and "Circle-Free" (*E* 60-1, *A* 76)

- 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 (either recurring, in the case of rational numbers, or non-repeating).
  - 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".]

110

110

The Complete Configuration (*E* 60, *A* 75)

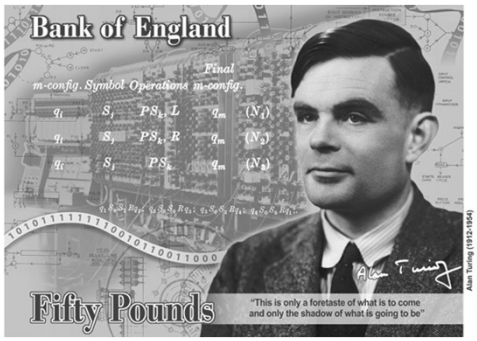
- We have already come across the terms:
  - "*m*-configuration" (the *state* of the machine)
  - "*configuration*" (i.e. *current state plus current symbol*)
- But Turing defines another similar term, to capture the complete state of a computation at any time:

  "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."

111

111

3.2 Examples of Turing Machines



112

112

§ 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:

  0 1 0 1 0 1 0 1 0 1 …

- In the full form (*E* 61, *A* 81), this requires 4 states (starting from "b" for "begin" – in an ordinary font here, rather than Turing's hard-to-read Fraktur):

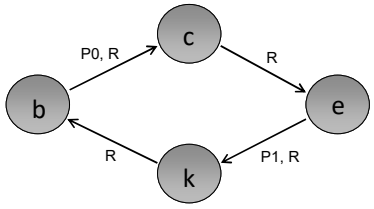
  b Prints a "0" and moves right into state c
  c Moves right into state e
  e Prints a "1" and moves right into state k
  k Moves right into state b

113

113

Configuration *Behaviour*

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



114

114

Turing on Computability and Intelligence

Simplifying the Tables

- Turing points out that machine tables can be greatly simplified if we allow a single transition to perform multiple operations (*E* 61, *A* 84):

m-config.	*symbol*	*operations*	*final m-config.*
	None	P0	b
b	0	R, R, P1	b
	1	R, R, P0	b

115

115

Turing's Example Machine II

- Turing then gives a far more complex example, which computes the sequence:

  0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 …

  (Petzold gives detailed explanation at *A* 85-90)
  - This is irrational, probably transcendental.
  - Note Turing's convention of marking the left-hand end of the tape (i.e. printing on the first *F*- and *E*- squares) two "schwa" characters: "ƏƏ".
  - Note also that two successive blanks always signify the right of the printed part of the tape.

116

116

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. (*E* 62, *A* 92):

    b : Ə Ə o 0 0 : Ə Ə q 0 0 : …
  - This format is flagged as "(C)" and referred to later, in § 6 of the paper (*E* 68, *A* 144).

117

117

Getting Used to Turing Machines

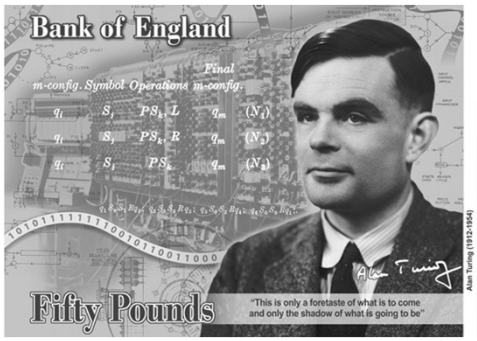
- To familiarise yourself with how Turing machines work, see Chapter 6 ("Addition and Multiplication") of Petzold's *Annotated Turing*.
- This gives two nice examples (not Turing's):
  - A machine to generate all the positive integers in sequence (*A* 99-100);
  - A machine to calculate the square root of 2 (*A* 100-108).

  Both of these are built into the *Turtle System*'s Turing machine simulator (in Python or Pascal).

118

118

3.3 The Tape-Editing Subroutines



119

119

§ 4 Abbreviated Tables (*E* 63, *A* 113-5)

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

120

120

Turing on Computability and Intelligence

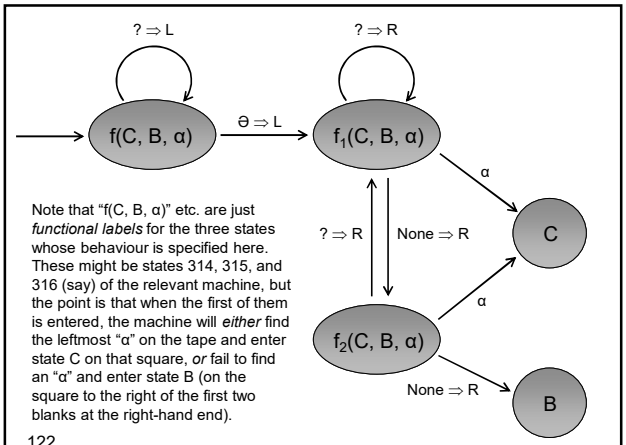
Turing's First *m*-function "Find"

m-config.	Symbol	Behaviour	Final m-config.
	ə	L	$f_1(C, B, \alpha)$
f(C, B, α)	not ə	L	f(C, B, α)
	None *	L	f(C, B, α)
	α		C
$f_1(C, B, \alpha)$	not α	R	$f_1(C, B, \alpha)$
	None	R	$f_2(C, B, \alpha)$
	α		C
$f_2(C, B, \alpha)$	not α	R	$f_1(C, B, \alpha)$
	None	R	B

This searches for the leftmost α symbol. If found, state C is entered at that point; if not, state B is entered.

121

121



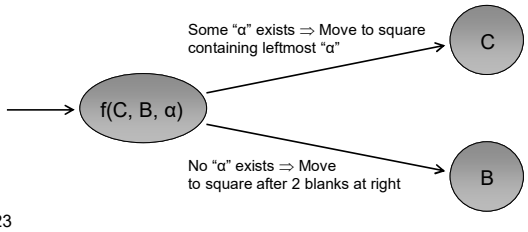
Note that "f(C, B, α)" etc. are just *functional labels* for the three states whose behaviour is specified here. These might be states 314, 315, and 316 (say) of the relevant machine, but the point is that when the first of them is entered, the machine will *either* find the leftmost "α" on the tape and enter state C on that square, *or* fail to find an "α" and enter state B (on the square to the right of the first two blanks at the right-hand end).

122

122

Summarising "Find"

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



123

123

"Erase" (3 arguments, erase once)

m-config.	Behaviour	Final m-config.
e(C, B, α)		$f(e_1(C, B, \alpha), B, \alpha)$
$e_1(C, B, \alpha)$	E	C

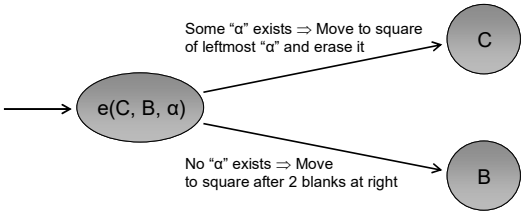
Turing (p. 118) defines two versions of this *m*-function, first with three arguments (shown above), then with two. On going into state e(C, B, α), the machine transitions into state $f(e_1(C, B, \alpha), B, \alpha)$, which searches for the leftmost "α" and enters state $e_1(C, B, \alpha)$ on the square containing that "α" (or ends in state B if there is no "α"). Then it erases that square (action E) and enters state C.

124

124

Summarising "Erase"

Here is the simplified diagram for e(C, B, α):



This *m*-function will now be used to define e(B, α).

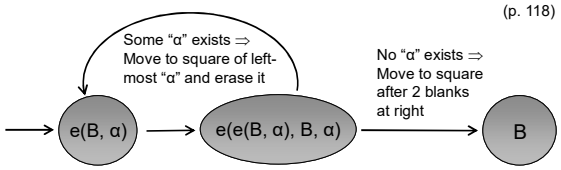
125

125

"Erase" (2 arguments, repeated erase)

m-config.	Behaviour	Final m-config.
e(B, α)		e(e(B, α), B, α)

(p. 118)



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

126

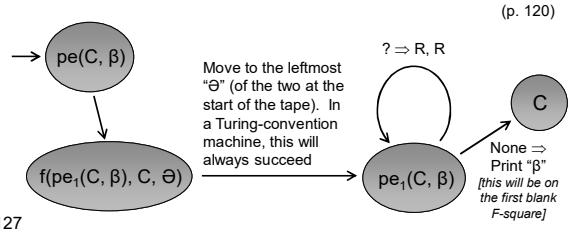
126

Turing on Computability and Intelligence

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

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
pe(C, β)			f(pe$_1$(C, β), C, Ə)
pe$_1$(C, β)	Any	R, R	pe$_1$(C, β)
pe$_1$(C, β)	None	Pβ	C

(p. 120)



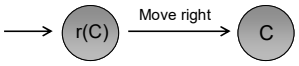
127

127

"Left", "Right", f′ and f″ *m*-functions

m-config.	*Behaviour*	*Final m-config.*
l(C)	L	C
r(C)	R	C

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



f′(C, B, α)		f(l(C), B, α)
f″(C, B, α)		f(r(C), B, α)

f(r(C), B, α) finds the left-most "α" then enters state r(C), thus moving right and entering state C. So f″(C, B, α) is similar in effect to f(C, B, α), except that it moves right after finding the leftmost "α".

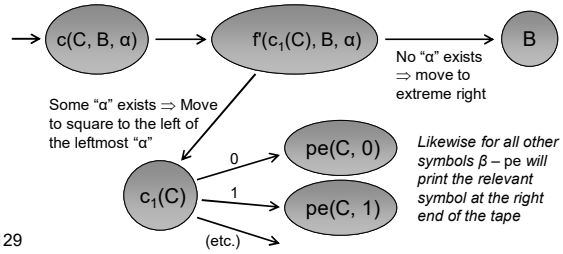
128

128

The "Copy" *m*-function

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
c(C, B, α)			f′(c$_1$(C), B, α)
c$_1$(C)	β		pe(C, β)

(p. 121)



129

129

The "Compare" *m*-function

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
cp(C, A, E, α, β)			f′(cp$_1$(C,A,β), f(A,E,β), α)
cp$_1$(C, A, β)	γ		f′(cp$_2$(C,A,γ), A, β)
cp$_2$(C, A, γ)	γ		C
cp$_2$(C, A, γ)	not γ		A

Turing (p. 123) explains the behaviour of this *m*-function thus: "The first symbol marked α and the first marked β are compared. If there is neither α nor β, → E.* If there are both and the symbols are alike, → C. Otherwise → A."

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

130

130

A Summary of "Subroutines"

- The main tape-manipulating "subroutines" defined in this way in § 4 are:

f	find first "α" (f'/f" then move left/right)
e	erase first "α" (or all "α"s, or all markers)
pe	print "β" in the first blank *F*-square
c	copy into the first blank *F*-square the first symbol marked with an "α"
ce	copy at the end all the symbols marked with an "α", then erase the "α"s
re	replace first "α" (or all "α"s) with "β"

131

131

cr	copy at the end all the symbols marked with an "α" (without erasing the "α"s)
cp	compares the first symbol marked "α" with the first symbol marked "β", entering different next states depending on the outcome
cpe	likewise, but erase markers if similar
	(or compare two sequences …)
g	find last "α" (the paper calls this "q", but "g" seems intended – see p. 124)
pe$_2$	print 2 characters in two *F*-squares
ce$_2$, ce$_3$	copy to the end symbols marked α, β (and γ), erasing the symbols

132

132

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 4

The 1936 Paper's Inspiration and Strategy, and the Universal Turing Machine



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

133

What Was the Primary Inspiration behind the Turing Machine?



(a) David Hilbert (1862-1943)

Posed the "decision problem" or "*Entscheidungsproblem*" in 1928



(b) A human "computer"

134

(a) "… With an Application to the *Entscheidungsproblem"*

- § 11 of the 1936 paper, "Application to the *Entscheidungsproblem"* (*E* 84-7, *A* 260-79), proves that Hilbert's decision problem *cannot be solved*, by presenting a family of formulae of predicate logic, for which – Turing argues – there is no general effective method of deciding whether they are true or false.
- It seems plausible that he set out to design his "computing machines" as models of effective computation, precisely to establish this momentous result with which his paper ends: "to show that the *Entscheidungsproblem* cannot be solved" (*E* 87, *A* 277).

135

"A Turing machine is an abstract model of human computation"

- "He … considers the actions of an abstract human being … Turing easily shows that the behavior of the computor can be exactly simulated by a Turing machine" (Gandy 1988, pp. 81-2).
- "the Turing machine [like] the 'Turing Test', was … based on a principle of imitation … modelled by considering what a human being could do when following a definite method." (Hodges 2009, p. 14).
- "A Turing machine is an abstract mathematical model of a human clerk. … Turing wanted to know which mathematical tasks could … be performed by a human clerk." (Sprevak 2017, p. 281).

136

A Third Hypothesis

- The primary inspiration for the Turing machine was neither of these:
  - a) Tackling Hilbert's *Entscheidungsproblem*
  - b) Modelling human algorithmic thinking

  but rather:
  - c) Exploring the logic of the Richard Paradox, with *computability* taking the place of *definability*

For details, see Peter Millican, "Alan Turing and Human-Like Intelligence", in Stephen Muggleton and Nicholas Chater (eds), *Human-Like Machine Intelligence*, Oxford University Press, 2021, pp. 24-51; the pre-publication version is available at www.millican.org/papers/2020Turing.pdf.

137

A Subtle Hint of Turing's Inspiration?

8. *Application of the diagonal process.*

It may be thought that arguments which prove that the real numbers are not enumerable would also prove that the computable numbers and sequences cannot be enumerable*. It might, for instance, be thought that the limit of a sequence of computable numbers must be computable. This is clearly only true if the sequence of computable numbers is defined by some rule.

* *Cf.* Hobson, *Theory of functions of a real variable* (2nd ed., 1921), 87, 88.

- From § 8 of the paper (*E* 72, *A* 173). Ernest Hobson was Professor of Mathematics at Cambridge until 1931, the year Turing arrived. The footnote seems to refer to Cantor's diagonal argument, but surprisingly, it does not!

138

Turing on Computability and Intelligence

Hobson's Unexpected Text

- Pages 87 and 88 of Hobson's textbook instead discuss the notion of *finitely definable numbers*:

  "By means of a given stock of … words and symbols it is possible to define only an enumerable set of [real numbers], … On this ground it has been maintained by König that the [real numbers] fall into two classes: … numbers capable of finite definition, and those … inherently incapable of finite definition."

- Hobson strongly disagrees with König's claim. A footnote in this second sentence refers to two papers of 1905, by König and Richard, both of which present *paradoxes of definability*.

139

139

"On Computable Numbers …"

- If we replace the notion of *definability* by that of *computability*, we get straight to the heart of Turing's treatment of *computable numbers*, which he does claim to be a distinct class (contrasting with Hobson's view of definable numbers).
- Turing's early interest in the definable/computable distinction is apparent in paragraph 2 of the paper:

  "The computable numbers do not … include all definable numbers, and an example is given of a definable number [defined through diagonalisation] which is not computable" (*E* 58, *A* 66).

- To justify his claim, however, Turing must show how the paradox of Richard can be avoided …

140

140

Imagining Turing's Train of Thought

"Suppose there were some systematic way of defining real numbers between 0 and 1 *by specifying methods for computing them*, calculating and printing their digits (either as decimals or binimals), on and on forever.

If such methods are enumerable, then it ought to be possible to order and iterate through them, calculating the initial digits of each such *computable number* in turn, combined with Cantorian *tweaking* to generate – digit by digit – a 'diagonal' number which logically cannot itself be one of the computable numbers.

Yet if this can be done automatically, it would seem that the diagonal number is being computed, and that is paradoxical! So where exactly does it fail?"

141

141

The Point of the Turing Machine

- The Turing machine – as conceived in the 1936 paper – does exactly what is required to follow through this thought-experiment:
  - A machine table systematically defines how the digits of a number are to be computed.
  - A "circle-free" machine is one that carries on generating [binary] digits indefinitely, from left to right on a potentially infinite tape, *and thus succeeds in defining a computable number*.
  - Machine tables are reducible to a simple linear form which can be printed on a tape, and thus to an integer which makes them enumerable.

142

142

Iteration, Mimicking, Checking

- The thought-experiment also requires a mechanism for controlling iteration through these *machine integers*, unpacking them into linear instructions, then mimicking their digit generation and tweaking etc.
- Crucially, this mimicking process requires creation of a *universal machine*, which follows the instructions of the machine which has been unpacked onto the left-hand side of the tape, to generate an appropriately tweaked digit on the right-hand side of the tape.
- However mimicking should only be done on *circle-free* machines, which succeed in defining a computable number – so this must be checked for.

143

143

Requirements for a Richard-Style Paradox:

A. (1) A way of representing Turing machine rules as symbols on a tape; (2) A method of translating sets of such rules into machine integers, and vice-versa.

B. A way of checking when a set of Turing machine rules does indeed generate a computable number (i.e. the machine is "circle-free", printing digits indefinitely).

C. A "universal machine" which can mimic the execution of any given Turing machine, generating the same sequence of digits (i.e. the same computable number).

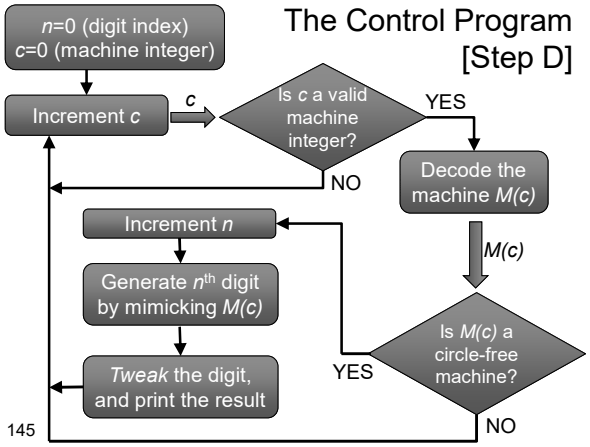
D. A control program that can generate the diagonal number by mimic-executing the encoded Turing machines in sequence, finding the relevant digit and *tweaking* it (so 0 becomes 1 and others become 0).

144

144

Turing on Computability and Intelligence

The Control Program [Step D]



145

145

The Richardian Logic of Turing's Paper

§5 "Enumeration of Computable Sequences"
Explains **Step A** – translating machines into sequences of symbols and ultimately single integers (and back).

§6 "The Universal Computing Machine"

§7 "Detailed Description of the Universal Machine"
Explains **Step C** – how to create a "universal machine" which can mimic the execution of any given machine, and generate the same sequence of digits.

§8 "Application of the Diagonal Process"
Explains why the prior **Step B** is essential to generating the diagonal number, and "applies the diagonal process", drawing appropriate conclusions.

146

146

§ 5 Enumeration of Computable Sequences [Step A]

- Turing explains (*E* 66, *A* 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 (S_0, S_1, S_2 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.

147

147

- We have a line of text, consisting of quintuples (e.g. "$q_1 S_2 S_2 L q_2$") separated by semicolons, …
  - "In this description we shall replace q_i by the letter '*D*' followed by the letter '*A*' repeated *i* times, and S_j by '*D*' followed by '*C*' repeated *j* times." So:
    "q_1" becomes *"DA"*; "q_2" becomes *"DAA"*
    "S_0" becomes *"D"*; "S_1" becomes *"DC"*
  - This is the *standard description* (S.D.) of the machine (*E* 67, *A* 137)
  - "If finally we replace '*A*' by '1', '*C*' by '2', '*D*' by '3', '*L*' by '4', '*R*' by '5', '*N*' by '6', and ';' by '7' we … have a description of the machine [as] an arabic numeral. The integer represented by this numeral may be called a *description number* (D.N.) of the machine."

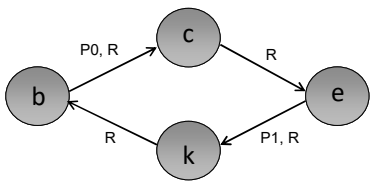
148

148

Finding a Machine's Description Number: Turing's Example (*E* 68, *A* 138-40)

"Let us find a description number for the machine I of § 3."

This is the machine (E 61, A 81, slide 114) that prints out the binimal "0 1 0 1 0 1 …":



149

149

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

"When we rename the *m*-configurations [and symbols] its table becomes:

q_1	S_0	PS_1, R	q_2
q_2	S_0	PS_0, R	q_3
q_3	S_0	PS_2, R	q_4
q_4	S_0	PS_0, R	q_1

150

150

Turing on Computability and Intelligence

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

q_1	S_0	PS_1, R	q_2
q_2	S_0	PS_0, R	q_3
q_3	S_0	PS_2, R	q_4
q_4	S_0	PS_0, R	q_1
q_1	S_1	PS_1, 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).

151

151

- 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 "circle-free" machines to computable numbers is *surjective* (every computable number must correspond to at least one such machine) but it is not *injective*.

152

152

Having explained his two illustrative tables, Turing continues (E 68, A 139-40):

"Our first standard form would be

$q_1\ S_0\ S_1\ R\ q_2\ ;\ q_2\ S_0\ S_0\ R\ q_3\ ;\ q_3\ S_0\ S_2\ R\ q_4\ ;$
$q_4\ S_0\ S_0\ R\ q_1\ ;\ .$

The standard description is

DADDCRDAA;*DAADDRDAAA*;
DAAADDCCRDAAAA;*DAAAADDRDA*;"

["q_1"⇒"DA", "S_0"⇒"D", "S_1"⇒"DC", "q_2"⇒"DAA", "q_3"⇒"DAAA", etc.]

153

153

Now, as Turing explained earlier, we simply replace 'A' by '1', 'C' by '2', 'D' by '3', 'L' by '4', 'R' by '5', 'N' by '6', and ';' by '7', to yield the corresponding description number …

"A description number is
3133253117311335311173111332253111731111335317

and so is
313325311731133531173111332253111731111335317311731323253117"

The second number here is for the machine with the "irrelevant" final quintuple.

154

154

The Computable Numbers Are Enumerable

- "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." (*E* 68, *A* 138)
- This is a significant result: it follows that "nearly all" of the real numbers in Cantor's universe are not computable.

155

155

Iterating through Possible Machines

- Now we have:
  - (A1) A method of encoding Turing machine rules as sequences of symbols printed on a tape (i.e. the *standard description*);
  - (A2) A method of translating such sequences of symbols into – and back from – a *machine integer* or *description number* (plus a fairly obvious effective method of checking that the number is a valid representation of a set of machine rules).
- Hence we can now create a machine which will count through the positive integers (1, 2, 3, etc.), checking in turn whether each of them is a valid machine integer, and if so, translating it into the standard description as printed on the left of a tape. So we can iterate in turn through all possible Turing machines.

156

156

Turing on Computability and Intelligence

Checking Whether a Machine is "Circle-Free" [Step B]

- *A "circle-free" machine is one that generates a computable number*, in the form of a potentially infinite sequence of [binary] digits.
  - The digits are interpreted as following after "0."
  - Non-digits are irrelevant here – they can be ignored.
  - Most machines are not circle-free in this way: they fail to continue printing further digits indefinitely.
- To create our paradox, we need to be able to design a Turing machine function *CF* which can test whether any given Turing machine representation *T* is *circle-free* in this sense. (See *E* 72, *A* 177 for this point.)

157

157

Mimicking an Encoded Machine [Step C]

- So far, we have a machine that iterates through all possible Turing machine representations in turn, and tests whether they are "circle-free" (in the sense of generating a computable number).
- To create our Richard-style paradox, we now want to generate each such computable number up until the n^{th} digit (before *tweaking* that digit). So we need a machine *U* which can mimic any other Turing machine *M*, based on its encoded form as printed on the machine tape.
- This – explained in § 6 and § 7 of the 1936 paper – is one of its greatest innovations – the ***Universal Turing Machine*** *– a machine (or function) which can mimic the execution of any machine whose specification it is given*.

158

158

Brief Outline of The Universal Machine

- Machine *U* mimics machine *M* as follows:
  - *M*'s specification (i.e. encoded rules in the form of a standard description) is printed on the left of the tape.
  - *U* starts by printing at the right of the tape the initial "complete configuration" of *M*'s computation (i.e. *M*'s tape contents, *M*'s state, and its current square).
  - *U* then takes note of *M*'s current state and symbol, moves left until it has found the appropriate rule (within the encoding of *M*'s rules), and applies that rule to generate and record *M*'s new complete configuration (at the right of the tape). This repeats for each step.
  - As it does this, *U outputs the same digits – and hence the same computable number – as M would output.*

159

159

§ 6 The Universal Machine

- In § 6 (*E* 68-9, *A* 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 $\mathscr{M}$, then $\mathscr{U}$ will compute the same sequence as $\mathscr{M}$."
- As Petzold remarks, Turing starts rather oddly:

  "Let us first suppose that we have a machine $\mathscr{M}'$ which will write down on the *F*-squares the successive complete configurations of $\mathscr{M}$."

160

160

Recalling Machine II

- Turing now refers back to Machine II of § 3 (*E* 62, *A* 87-92, slide 116), which generates the sequence "0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 …".
  - This again seems odd, because that machine was defined with multiple operations in most lines of the table (cf. *E* 61, *A* 84, slide 115), but at this point in the paper Turing is restricting his attention to simpler "standard form" machines (*E* 66-7, *A* 132, slide 114).
  - Ignore this detail, because the process Turing is about to illustrate using Machine II and format "(C)" will also work with standard form machines.

161

161

Generating Complete Configurations

- Recall from slide 117 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. (*E* 62, *A*. 92):

  b : Ə Ə o 0 0 : Ə Ə q 0 0 : … (C)
- At *E* 68-9, *A* 144-6, Turing returns to and modifies this format by replacing the state codes (b o q) with "DA", DAA", and "DAAA", blanks by "D", "0" by "DC", "1" by "DCC", and "Ə" by "DCCC", to fit with the *standard description*.

162

162

Turing on Computability and Intelligence

Machine $\mathscr{M}'$

- The result of these substitutions into (C) is:

  "*DA* : *DCCCDCCCDAADCDDC* :
  DCCCDCCCDAAADCDDC : … (C_1)

  (This is the sequence of symbols on *F*-squares.)"

  – Turing (*E* 69, *A* 146) means that the machine $\mathscr{M}'$, 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).
- He remarks that "if $\mathscr{M}$ can be constructed, then so can $\mathscr{M}'$." It would operate by referring back to a copy of the S.D. of $\mathscr{M}$, written on the tape.

163

163

How Machine $\mathscr{M}'$ Will Work

- $\mathscr{M}'$ will print out, in sequence, the complete configurations that $\mathscr{M}$ would produce.
- At each stage, it will have a record of the last complete configuration (at the right of the tape), and a record of $\mathscr{M}$'s rules (in the form of the standard description) at the left of the tape.
- It will shuttle back and forth, checking the latest "configuration" (i.e. state and symbol) from the right, then finding the rule that this matches at the left, then moving back to build the next complete configuration accordingly on the right.

164

164

Identifying the Configuration

- Here the configurations in (C_1), as generated by [non-standard] Machine II (*E* 62, *A* 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 *E* 70, *A* 151.

165

165

Identifying the Matching Rule

- We can underline the "trigger" configurations in the standard description of Machine I (*E* 61, 68; *A* 81, 139-40) *(we can't do this for the non-standard Machine II):*

  "*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
DA	*D*	*DC*	*R*	*DAA*

166

166

Outputting Figures

- In format (C_1) [slides 163, 165; *E* 69, *A* 146], no actual figures (i.e. digits) appear as the successive configurations are built up at the right of the tape:

  "*DA* : *DCCCDCCCDAADCDDC* :
  DCCCDCCCDAAADCDDC : … (C_1)"

  ə ə q 0 - 0

- As Turing explains (*E* 69, *A* 147-8), "We may correct this by printing between each successive pair of complete configurations *the figures which appear in the new configuration but not in the old*. Then …"

167

167

DA : *DCCCDCCCDAADCDDC* :
DCCCDCCCDAAADCDDC : … (C_1)

becomes:

DA : 0 : 0 : *DCCCDCCCDAADCDDC* :
DCCCDCCCDAAADCDDC : … (C_2)

- So now, machine $\mathscr{M}'$ 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 $\mathscr{M}'$ will contain "*D*", "*C*", "*A*", or ":", with the occasional "0" or "1" interspersed.

168

168

Turing on Computability and Intelligence

Achieving Universality

- Machine $\mathscr{M}'$ 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*.

169

169

§ 7 Detailed Description of the Universal Machine

- § 7 describes Turing's Universal Machine in detail, making use of many of the "subroutine" functions whose skeleton tables were in § 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.

170

170

- 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 (*E* 69, *A* 149).
  - For example the *m*-function "e(anf)" is used, and this requires also a state "e_1(anf)" (*E* 69-70, *A* 150).
- "When $\mathscr{U}$ is ready to start work the tape running through it bears … the symbol ə on an *F*-square and again ə 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." (*E* 70, *A* 150)
  - Note Turing's use of the word "instructions" here (cf. *A* 82).

171

171

- At *E* 70 / *A* 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 168), it also needs *E*-square marker symbols "*u*", "*v*", "*w*", "*x*", "*y*", and "*z*".
- At *E* 70 / *A* 152, Turing specifies a skeleton table for one last *m*-function "con". This is used to identify – and mark with a specified symbol α (e.g. "*x*") – the configuration closest to the right of the current scanned square. See *A* 152-3 for Petzold's explanation and an example.

172

172

How $\mathscr{U}$ works

1. It starts off printing ":*DA*" (representing a blank tape in the initial state) on the *F*-squares after the "::" (*E* 70, *A* 153). It then goes into a ***big loop*** …
2. It finds the last colon on the tape (which precedes the last complete configuration) and marks the configuration (i.e. the state and scanned symbol) with "*y*" (*E* 70, *A* 154).
3. Now it goes into ***another loop*** to identify the relevant instruction. It goes back to find the last semicolon not (yet) marked with "*z*"; it then marks this with "*z*", and marks the configuration following (i.e. the trigger of the relevant rule) with "*x*".

173

173

4. Next (*E* 71, *A* 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, …", *E* 71, *A* 156). 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 (*E* 71, *A* 157).
7. "0" or "1" is printed at the end of the tape, if such a print instruction is present (*E* 71-2, *A* 159).
8. The resulting complete configuration is written at the end of the tape, "carrying out the marked instructions" (*E* 72, *A* 160). ***Back to the big loop***!

174

174

Turing on Computability and Intelligence

§ 8 Application of the Diagonal Process

- Turing's reference to "the diagonal process" (not previously mentioned) tends to confirm that he has had this Richardian denouement in mind all along.
- In § 5 he demonstrated that the computable numbers are enumerable, but he now remarks (with reference to Hobson's book) that Cantor's diagonal argument might seem to prove otherwise.
- If the computable [binimal] numbers are indeed enumerable, then we should be able to form a list that contains them all.

175

175

- Suppose, then, that this is the list:

1: $0.a_{11}a_{12}a_{13}a_{14}a_{15}a_{16}a_{17}a_{18}\ \ldots$
2: $0.a_{21}a_{22}a_{23}a_{24}a_{25}a_{26}a_{27}a_{28}\ \ldots$
3: $0.a_{31}a_{32}a_{33}a_{34}a_{35}a_{36}a_{37}a_{38}\ \ldots$
4: $0.a_{41}a_{42}a_{43}a_{44}a_{45}a_{46}a_{47}a_{48}\ \ldots$
… …

- Consider now the number β (*E* 72, *A* 174) that we get if we take the digits in order down the long diagonal of this list, changing "0" into "1" and "1" into "0". That number must differ from every number in the list, so if Turing is correct, it cannot be a computable number.

176

176

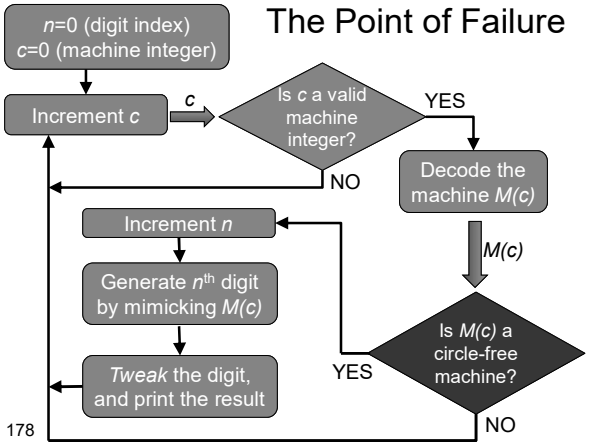
What Prevents Computation of β?

- Yet how can β 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 1st, 2nd, … n^{th} … machine until we get the n^{th} digit, swap "0" for "1" and continue?
- Why won't this give a way of computing β, and thus refute Turing's claim of enumerability?

177

177

The Point of Failure



178

178

Achieving a Paradox?

- If this flow diagram works smoothly as described, then we have the paradox of a *mechanical computing process for generating a number which, by definition, should be uncomputable*.
- So it must fail somewhere! But where?
  - Turing's way of defining machine integers was clear and unambiguous, making them easily translatable into machine encodings (and vice-versa).
  - His Universal Machine was well-defined, and clearly capable of mimicking any suitably specified machine.
  - Setting, incrementing, and counting through integers, and *tweaking* digits, are all straightforward.

179

179

- To complete the diagonal process and compute β, we have to be more discriminating than just to iterate through the integers that are description numbers of Turing machines. A computable number is generated only by a *circle-free* machine, so we must compute whether or not any given integer is the description number of a *circle-free* machine.
- Since computable construction of the diagonal number β would contradict the enumerability of the computable numbers, and since the latter has already been proved, Turing concludes that there cannot be any "general process" for "finding out whether a given number is the D.N. of a circle-free machine ... in a finite number of steps" (*E* 72, *A* 177-8).

180

180

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 5

Undecidable Problems, and Settling Hilbert's Entscheidungsproblem

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



181

Turing's Conclusion So Far

- As we have seen, Turing concludes early in § 8 (*E* 72, *A* 177) that there can be no general method for calculating, in a finite number of steps, whether a given integer is, or is not, the description number of a *circle-free* machine (i.e. a machine that will carry on printing further digits indefinitely).
- His initial argument is that if there were such a general method, then it could be applied within "the diagonal process" of § 8 and generate a contradiction involving the number β (exactly analogous to Richard's paradox, but involving the *computability* rather than *definability* of β).

182

182

Where the Attempt Fails

- But Turing remarks that the reader might feel "there must be something wrong" with this proof, which proceeds indirectly by contradiction. He then explains more directly *where* the attempt to construct β fails (*E* 72-3, *A* 178-83).
- It breaks down when our would-be machine $\mathscr{H}$ comes to check its own description number *N*. For then in order to determine the N^{th} digit that it is supposed to output, it must *first* discover what the N^{th} digit of machine number *N* (i.e. $\mathscr{H}$ itself) would be. So it never finds the answer.

183

183

More "Diagonal" Arguments

- Turing's Richardian argument here is reminiscent both of Cantor's diagonal arguments, and also Gödel's proof (which involves a formula that indirectly asserts its own unprovability).
- Other "diagonal" arguments can be used to prove further analogous results regarding the behaviour of computer programs, and Rice's Theorem (1951) states the generalisation that *all non-trivial semantic properties of programs are undecidable*.
- The *Halting Problem* is the most famous of these results, and can be proved surprisingly easily …

184

184

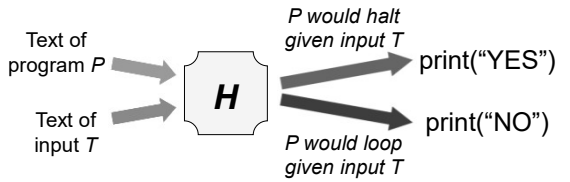
The Halting Problem

- 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?*
- *A Turing-style argument by contradiction can prove very quickly that H is impossible* (though Turing himself did not discuss this).

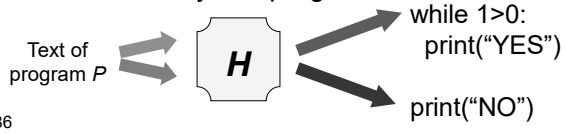
185

185

- Suppose we have such a procedure *H:*



- We now modify that program:



186

186

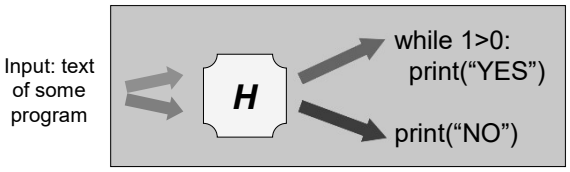
Turing on Computability and Intelligence

- Clearly if the initial program is possible, then the modified version must be possible too, because the changes are so simple:
  - The *input* process has modified so that instead of taking *both* a program file *P and* a text file *T* as input, it uses the program file in place of *T* – hence its verdict is based on whether or not the specified program file *P* halts when given its own text as input. (The modified program could simply start by duplicating file *P* and taking the duplicate as *T*.)
  - The *output* process has also been very simply modified, so that if the "it would halt" test comes out positive, the program goes into an infinite loop.

187

187

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.

188

188

An Important Lemma

- 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)." (*E* 73, *A* 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.

189

189

- Suppose we have our 0-testing machine $\mathscr{E}$, and we are considering the behaviour of some machine $\mathscr{M}$.
- We then consider machines $\mathscr{M}_1$, $\mathscr{M}_2$, … $\mathscr{M}_n$, …, 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. "ϕ"). It is possible *mechanically* to create in turn the standard decriptions of these machines from that of $\mathscr{M}$, by adding more states.
- Now we create another machine $\mathscr{G}$ from $\mathscr{E}$, which operates by mechanically creating these SDs and testing $\mathscr{M}$, $\mathscr{M}_1$, $\mathscr{M}_2$, … in turn, outputting a "0" each time if and only if $\mathscr{E}$ would decide that the tested $\mathscr{M}$-machine would <u>*never*</u> generate a "0".
- So, for example, if $\mathscr{M}$ would print 4 "0"s, we get:

190

190

Machine	output 1	output 2	output 3	output 4	output 5	$\mathscr{G}$ outputs
$\mathscr{M}$	0	0	0	0	-	-
$\mathscr{M}_1$	ϕ	0	0	0	-	-
$\mathscr{M}_2$	ϕ	ϕ	0	0	-	-
$\mathscr{M}_3$	ϕ	ϕ	ϕ	0	-	-
$\mathscr{M}_4$	ϕ	ϕ	ϕ	ϕ	-	0
$\mathscr{M}_5$	ϕ	ϕ	ϕ	ϕ	-	0

- $\mathscr{G}$ 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.

191

191

Deducing the Lemma

- 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" (*E* 74, *A* 186).
- But then, "By a combination of these processes we have a process for determining whether $\mathscr{M}$ prints an infinity of figures, *i.e.* we have a process for determining whether $\mathscr{M}$ is circle-free. There can therefore be no machine $\mathscr{E}$."

192

192

Turing on Computability and Intelligence

§ 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).

193

193

Why Wait Until § 9 to Argue for the Generality of Turing Machines?

1. The paper starts from the notion of *computable numbers*, as captured by the Turing machine, and takes exactly the steps that are needed to run the logic of the Richard Paradox in these terms.
2. At § 8, this reveals a generally uncomputable problem: whether a given Turing machine is "circle-free" and thus generates a computable number.
3. Only *after* he had discovered this problem, I suggest, Turing realised that he could use it to prove the unsolvability of the *Entscheidungsproblem*, if he could identify a related uncomputable property that is expressible in terms of predicate logic – this is achieved with the lemma of § 8.

194

194

4. But to settle the *Entscheidungsproblem*, his machines must encompass all *effective methods* of computation.
5. So Turing added § 9, arguing (1) that his machines can perform any computation that can be performed by a human following a definite method, and (2) that they are equivalently powerful to predicate logic. He also added § 10, demonstrating (3) the power of his machines to calculate a wide range of number types.
6. An appendix proved his machines equivalent to Church's lambda calculus and Gödel's general recursion, thus establishing the *Church-Turing Thesis*.
7. Having persuasively argued that his machines provide a general model of effective computation, Turing could end his paper in § 11 by demonstrating his negative resolution of Hilbert's *Entscheidungsproblem.*

195

195

Turing's First Argument for the Church-Turing Thesis

- Turing (§ 9, *E* 74-5, *A* 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 204 below).
- The first type of argument (*E* 75-7, *A*190-4) maintains that a Turing machine can compute any number that would be computable by a human following a definite process. Turing anticipated this back in § 1, at *E* 59, *A* 68 and the argument was covered in Lecture 3 (see slides 94-100).

196

196

Turing's Second Type of Argument

- Turing's second type of argument – showing that various different ways of understanding computability are equivalent – has almost certainly been the most influential.
  - Here he first argues (*E* 77-8, *A* 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."

197

197

A "British Museum Algorithm"

- This kind of machine $\mathscr{K}$ has been called a "British Museum" algorithm (Petzold, *A* 220-1), generating in turn all provable formulae of a system of axioms.
  - The name suggests that using such an algorithm is like searching through the British Museum's huge collection item by item, hoping eventually to find the particular proof one wants.
- Turing then briefly points out (*E* 78-9, *A* 229) that any Turing machine can be defined in terms of predicate formulae. (This is demonstrated in far more detail in § 11.)

198

198

Turing on Computability and Intelligence

An Important Equivalence

- 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." (*E* 78, *A* 229)
- So he has thereby sketched "A proof of the equivalence of two definitions" of computable numbers, as he anticipated at *E* 75, *A* 190.

199

199

Turing is "Scooped" by Church

- 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.

200

200

Turing's Appendix

- Turing's Appendix proves in outline "The theorem that all effectively calculable (λ-definable) sequences are computable [by a Turing machine] and its converse" (*E* 88, *A* 290)
  - See Petzold Chapter 15 for much more on the Lambda Calculus etc.; also *A* 325-31.
- In Volume 2 of *The Journal of Symbolic Logic* (1937), Turing's "Computability and λ-definability" provided a more rigorous proof. The new paper starts as follows:

201

201

"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 λ-definable functions of Church and the general recursive functions due to Herbrand and Gödel [in 1933] and developed by Kleene. It is shown [below] that every λ-definable function is computable and that every computable function is general recursive." (*A* 298)

- Church and Stephen Kleene had already (June 1936) proved that recursive and λ-definable functions coincide, so it indeed followed that all three notions are equivalent.

202

202

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 give a formal precisification of the informal notion have turned out to be equivalent.
  - This explains why the C-T Thesis is so widely endorsed, though its precise formulation and further implications remain controversial.

203

203

§ 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 (*E* 80-1, *A* 240-1; proofs at *E* 82-3 *A* 247-56)
    - Turing slips here, because "carries" can propagate back from later digits to earlier (Joel Hamkins, *Lectures*, 2020, pp. 212-3).
  - the root of a computable function that crosses zero is computable (*E* 81, *A* 244), as is the limit of any "computably convergent sequence" (*E* 81, *A* 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." (*E* 82, *A* 247)
  - π and *e* are computable (*E* 82, *A* 247)
  - "all real algebraic numbers are computable" (*E* 82, *A* 247).

204

204

§ 11 Application to the *Entscheidungsproblem*

- § 11 finally comes round to Hilbert's famous *Entscheidungsproblem*, widely seen as the inspiration for the entire paper (though these lectures have suggested that Turing came to this only *after* he had discovered an uncomputable problem, by following the Richardian reasoning).
- Turing proves that the *Entscheidungsproblem* 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".

205

205

- Turing refers to Hilbert's presentation of "the restricted functional calculus" (i.e. what we call first-order 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 **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 $\mathscr{U}$ is provable." (*E* 84, *A* 260)
- He explains (*E* 84, *A* 261-2) that this result is quite different from Gödel's, as we saw in the second lecture (slide 89).

206

206

From Machines to Predicate Logic

- As Turing remarks (*E* 84, *A* 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 (*E* 84-5, *A* 263-7).
- 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 (*E* 85, *A* 267). By the earlier lemma (slides 189-92), this will in general be undecidable.

207

207

Three Predicates: Scan/State/Symbol

- $I(x, y)$ *E* 84, *A* 264
  In the complete configuration x, the square y is scanned.
- $K_{q_m}(x)$ *E* 85, *A* 264
  In the complete configuration x, the m-configuration [i.e. state] is q_m.
- $R_{S_l}(x, y)$ *E* 84, *A* 263
  In the complete configuration x (of $\mathscr{M}$), the symbol on the square y is S_l.

208

208

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 *A* 265-6). To do so, he also needs a predicate for numerical succession (which he had introduced at *E* 78, *A* 225, in preparation for discussing Peano's axioms):
- $F(x, y)$ *E* 85, *A* 264
  y is the immediate successor of x (i.e. $y = x + 1$).

209

209

- Suppose our machine includes quintuple $\{q_i\ S_j\ S_k\ L\ q_l\}$ 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', square y will change to symbol S_k and the machine will move left (to scan square y') and into state q_l. Clearly $x' = x + 1$ and $y' = y - 1$, so we have:

  $$F(x, x') \ \&\ F(y', y)$$

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

Cycle x	$I(x, y)$	&	$K_{q_i}(x)$	&	$R_{S_j}(x, y)$
Cycle x'	$I(x', y')$	&	$K_{q_l}(x')$	&	$R_{S_k}(x', y)$
			↑ Scanned square	↑ State	↑ Symbol

210

210

Turing on Computability and Intelligence

- We also need to express that all squares other than y will contain the same symbol in cycles x and x'.
- This yields the following formula (abbreviated to "Inst $\{q_i\ S_j\ S_k\ L\ q_l\}$", *E* 85, *A* 265) to express that quintuple:

(x, y, x', y') $\forall\ x, y, x'$, and y'

$\{\ I(x,y)\ \&\ K_{q_i}(x)\ \&\ R_{S_j}(x,y)$ IF cycle x scan/state/symbol

$\&\ F(x,x')\ \&\ F(y',y)$ & $x'=x+1,\ y'=y-1$

$\rightarrow$ THEN

$(I(x',y')\ \&\ K_{q_l}(x')\ \&\ R_{S_k}(x',y)$ cycle x' scan/state/symbol

$\&\ (z)\ [\ F(y',z)$ & $\forall z$ [either $z=y$ (i.e. $y'+1$)

$\vee\ ([R_{S_0}(x,z) \rightarrow R_{S_0}(x',z)]\ \&$ or square z keeps

$[R_{S_1}(x,z) \rightarrow R_{S_1}(x',z)]\ \&\ \ldots])])\}$ the same symbol]

211

211

- The formulae for Inst $\{q_i\ S_j\ S_k\ R\ q_l\}$ and Inst $\{q_i\ S_j\ S_k\ N\ q_l\}$ – to deal with quintuples in which the machine moves right or stays on the same square – will differ accordingly.
- Thus Inst $\{q_i\ S_j\ S_k\ R\ q_l\}$ will be the formula (*E* 85, *A* 266):

(x, y, x', y') $\forall\ x, y, x'$, and y'

$\{\ I(x,y)\ \&\ K_{q_i}(x)\ \&\ R_{S_j}(x,y)$ IF cycle x scan/state/symbol

$\&\ F(x,x')\ \&\ F(y,y')$ & $x'=x+1,\ y'=y+1$

$\rightarrow$ THEN

$(I(x',y')\ \&\ K_{q_l}(x')\ \&\ R_{S_k}(x',y)$ cycle x' scan/state/symbol

$\&\ (z)\ [\ F(z,y')$ & $\forall z$ [either $z=y$ (i.e. $y'-1$)

$\vee\ ([R_{S_0}(x,z) \rightarrow R_{S_0}(x',z)]\ \&$ or square z keeps

$[R_{S_1}(x,z) \rightarrow R_{S_1}(x',z)]\ \&\ \ldots])])\}$ the same symbol]

212

212

- Inst $\{q_i\ S_j\ S_k\ N\ q_l\}$ involves no machine movement, but we still keep y' 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):

(x, y, x', y') $\forall\ x, y, x'$, and y'

$\{\ I(x,y)\ \&\ K_{q_i}(x)\ \&\ R_{S_j}(x,y)$ IF cycle x scan/state/symbol

$\&\ F(x,x')\ \&\ F(y,y')$ & $x'=x+1,\ y'=y+1$

$\rightarrow$ THEN

$(I(x',y)\ \&\ K_{q_l}(x')\ \&\ R_{S_k}(x',y)$ cycle x' scan/state/symbol

$\&\ (z)\ [\ F(z,y')$ & $\forall z$ [either $z=y$ (i.e. $y'-1$)

$\vee\ ([R_{S_0}(x,z) \rightarrow R_{S_0}(x',z)]\ \&$ or square z keeps

$[R_{S_1}(x,z) \rightarrow R_{S_1}(x',z)]\ \&\ \ldots])])\}$ the same symbol]

213

213

From Machine Description To (Undecidable) Zero-Print Statement

- Every quintuple in machine $\mathscr{M}$ can be represented by one of these long formulae – which we have abbreviated as Inst $\{q_i\ S_j\ S_k\ N\ q_l\}$, Inst $\{q_i\ S_j\ S_k\ N\ q_l\}$ or Inst $\{q_i\ S_j\ S_k\ N\ q_l\}$. We then take the *conjunction* of all these (*E* 85, *A* 267), and call that Des($\mathscr{M}$).
  - Thus Des($\mathscr{M}$) gives a complete *description*, in predicate logic, of the operation of $\mathscr{M}$'s machine table.
- Following the strategy explained in slide 207 above, Turing next uses 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.

214

214

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 are captured by *Peano's Axioms*, (recall slide 54) of which this is one version (see *A* 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(successor(x)) for all numbers x, then $P(x)$ is true for all numbers x.

215

215

- Turing's original paper slips up in defining the Peano conditions (*E* 78; *A* 226-7, 267-8). His correction in 1937 (*E* 94; *A* 268, 311) uses Q as an abbreviation for:

$$(x)(\exists w)(y,z)\left\{\begin{array}{c} F(x,w)\ \&\ \big(F(x,y) \rightarrow G(x,y)\big) \\ \&\ (F(x,z)\&G(z,y) \rightarrow G(x,y)) \\ \&\ \left[\begin{array}{c} G(z,x) \vee (G(x,y)\&F(y,z)) \\ \vee\ (F(x,y)\&F(z,y)) \rightarrow (\neg F(x,z)) \end{array}\right] \end{array}\right\}$$

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.

216

(But even this isn't ideal, e.g. it doesn't rule out branching.)

216

Turing on Computability and Intelligence

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

- The formula Un($\mathscr{M}$) for machine $\mathscr{M}$, which is in general undecidable, is (*E* 85, 94-5; *A* 268, 311):

$$(\exists u)\,[Q\ \&\ (y)R_{S_0}(u,y)\ \&\ I(u,u)\ \&\ K_{q_1}(u)\ \&\ \mathrm{Des}(\mathscr{M})\,] \rightarrow (\exists s)(\exists t)R_{S_1}(s,t)$$

  - Apart from Q and Des($\mathscr{M}$), 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.

217

217

Two Lemmas To Prove

- Now Turing sets out to prove two lemmas, connecting the provability of Un($\mathscr{M}$) with the appearance of S_1 ("0") on the tape when machine $\mathscr{M}$ runs:
  - Lemma 1 (*E* 85-7, *A* 269-76): "If S_1 appears on the tape in some complete configuration of $\mathscr{M}$, then Un($\mathscr{M}$) is provable".
  - Lemma 2 (*E* 87, *A* 276-77): "If Un($\mathscr{M}$) is provable, then S_1 appears on the tape in some complete configuration of $\mathscr{M}$".

218

218

- To prove Lemma 1, Turing formulates a sequence of formulae CC_0, CC_1, CC_2, etc., representing the sequence of complete configurations as $\mathscr{M}$ proceeds (*E* 85-6, *A* 271).
- Here he uses an earlier abbreviation (from *E* 85, *A* 268), with "A($\mathscr{M}$)" standing for:

$$Q\ \&\ (y)R_{S_0}(u,y)\ \&\ I(u,u)\ \&\ K_{q_1}(u)\ \&\ \mathrm{Des}(\mathscr{M})$$

- 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 A($\mathscr{M}$) & $F^{(n)} \rightarrow CC_n$ are provable" (*E* 86, *A* 272: base case *A* 272-3; induction step *A* 273-5).

219

219

- Lemma 1 follows because, assuming that "0" (i.e. S_1) does indeed appear on the tape in configuration n (say), then CC_n (which includes all relevant $R_{S_i}(n,y)$ conjuncts), must include $R_{S_1}(n,y)$ for some square y.
- Lemma 2 – that if Un($\mathscr{M}$) is provable, then S_1 appears in some complete configuration of $\mathscr{M}$ – must be true (*E* 87, *A* 277) from substitution of the relevant propositional functions in Un($\mathscr{M}$).
- Thus (*E* 87, *A* 277), referring back to the lemma of § 8, "We are now in a position to show that the *Entscheidungsproblem* cannot be solved".

220

220

- § 8 began by concluding that no Turing machine can be constructed which will reliably determine whether any specified Turing machine is "circle-free".
- The lemma of § 8 then inferred that no Turing machine can be constructed which will reliably determine whether any specified Turing machine will ever print a "0".
- But since that behaviour (printing a "0", or not) can be captured by a predicate logic formula Un($\mathscr{M}$), it follows that no Turing 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. (On the assumption that Turing machines do indeed capture all possible mechanical computational procedures.)

221

221

Summing Up the 1936 Paper

1. Turing probably invented his "computing machines" to investigate the notion of a *computable number*, and in particular, to see what would happen if Richardian-style paradoxical reasoning were run in those terms.
2. Following this path led Turing to discover that the question of whether a given computing machine is "circle-free" is, in general, uncomputable.
3. Turing saw that this would deliver a negative answer to Hilbert's *Entscheidungsproblem*, if only:
   a) He could identify a related property expressible in terms of predicate logic (as in the lemma of § 8);
   b) He could make the case for his computing machines as general models of computing (as in § 9 and § 10).

222

222

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 6

Turing's 1950 Paper: Raising the Question of AI



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

223

Useful Books on Turing 1950

- Jack Copeland (ed.), *The Essential Turing* (OUP, 2004) – contains Turing's main publications and broadcasts, with helpful commentary. *"ET"* for short.
- Jack Copeland, Jonathan Bowen, Mark Sprevak and Robin Wilson (eds), *The Turing Guide* (OUP, 2017) – has over 40 approachable papers on Turing's life and work (Part V is on AI and the Mind).
- 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 to the Turing Test.

224

224

Useful Books on AI and Searle

- Philosophy of AI
  - Jack Copeland, *Artificial Intelligence: A Philosophical Introduction* (Blackwell, 1993).
  - Melanie Mitchell, *Artificial Intelligence: A Guide for Thinking Humans* (Pelican, 2019).
  - 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).
  - John Preston and Mark Bishop (eds), *Views into the Chinese Room* (OUP, 2002).

225

225

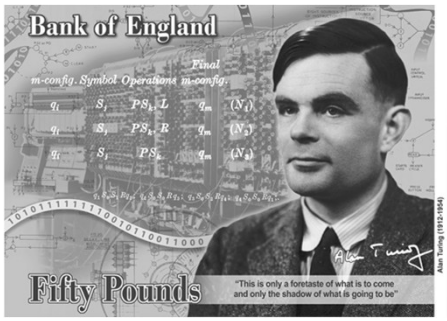
Useful Web Resources

- *The Stanford Encyclopedia of Philosophy* at http://plato.stanford.edu/ – see articles on "The Turing Test" and "The Chinese Room Argument".
- 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
- www.millican.org/research.htm contains two papers discussing the Turing Test, published in 2013 (Cooper and van Leeuwen collection) and 2021.

226

226

1. Quick Review of the 1950 Paper



Slides to be dwelt on in this quick review are starred: ★

227

227

★ "Computing Machinery and Intelligence" appeared in 1950, in the top philosophical journal, *Mind* (volume 59, pp. 433-60). It went on to became one of the most-cited philosophical papers ever!

- It does not, however, reach the standards of rigour that might be expected of a top philosophical paper. Robin Gandy, Turing's doctoral student, wrote:
  - "The 1950 paper was intended not so much as a penetrating contribution to philosophy but as propaganda. … He wrote this paper – unlike his mathematical papers – quickly and with enjoyment. I can remember him reading aloud to me some of the passages – always with a smile, sometimes with a giggle. Some of the discussions of the paper … load it with more significance than it was intended to bear."

(in Millican & Clark (eds) 1996, p. 125)

228

228

Turing on Computability and Intelligence

§ 1 The Imitation Game

- The paper starts rather oddly (p. 433, *ET* 441):

  "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."

- His "replacement" question is set in the context of what he calls the "Imitation Game".

229

229

- But why does a paper about "intelligence" start by posing its main question in terms of "thought"? It seems that – both in this paper and other writings on the topic – Turing uses "thought" as an informal synonym for "intelligence".
- The comment about using a statistical survey seems flippant. It might be appropriate to use a survey to find out what people in general mean by the word "liar", but having established its meaning, it's then a question of fact, not popular opinion, whether Boris (say) is, or is not, a liar.
- It seems rather peculiar to "replace" one question by another without explaining what the relationship between them is supposed to be.

230

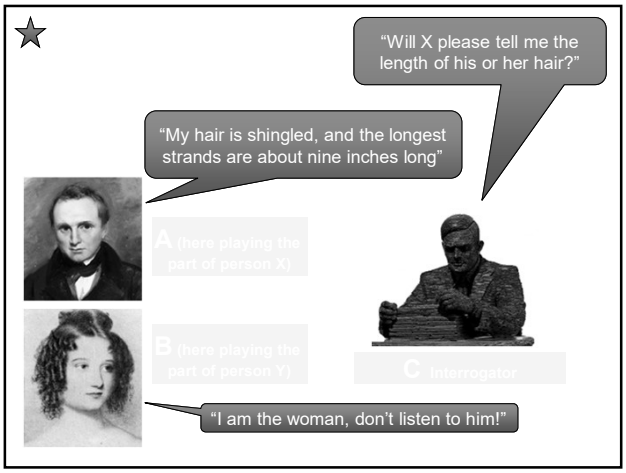
230

The Gendered Imitation Game

- Turing's introduces his "imitation game" in a form requiring an *interrogator* to guess the sex of two *witnesses* (whom he only knows as "X" and "Y", but knows to be of different sex), based on their answers to his questions. All questions and answers are communicated through written text.
- The man ("A") attempts by his answers to convince the interrogator ("C") that he is the woman – so he pretends to be female. The woman ("B") attempts to help the interrogator correctly identify her as the woman, by answering honestly. Introducing the game, Turing gives the following example:

231

231



232

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 …

233

233

The Computer Imitation Game

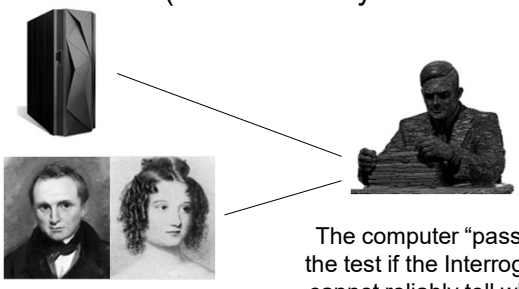
- "We now ask the question, 'What will happen when a machine takes the place of *A* [the deceitful man] in this game?' Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, 'Can machines think?'" (p. 434, *ET* 441)
- I believe, however, that Turing's quoted question should have been: 'What will happen when a machine takes the place of *A* in this game, with the part of *B* being taken by a man [or woman]?'.

234

234

Turing on Computability and Intelligence

The Turing Test
(as traditionally understood)



Can be either a male or female "man", i.e. human

The computer "passes" the test if the Interrogator cannot reliably tell which is the computer and which is the human.

235

- When Turing says "a machine takes the part of A", taken literally this would have *the machine pretending to be a woman, while the interrogator continues to be told that the two witnesses are a man and a woman*.
  - Some (e.g. Saygin et al.) suggest this is intended, but:
    - It is contradicted by Turing's text (slide 238 below).
    - It would be poor *as a test for intelligence*, because if one of the witnesses were actually a crude chatbot, the judge would have no basis for decision: an *idiot* or *obsessive* could just as well be male or female!
  - Copeland and Proudfoot suggest that Turing intends the gendered test as a *calibration for* the computer-imitates-human test. This fits the text on slide 234, but isn't suggested again, and Turing never discusses how well a man would do in the gendered test.

236

236

Saygin, Cicekli & Akman (2000)

> "One may ask why Turing designed the IG [Imitation Game] in such a peculiar manner. Why the fuss about the woman, the man, and the replacement? … On close examination, it can be seen that what Turing proposes is to compare the machine's success against that of the man, *not* to look at whether it 'beats' the woman in the IG. … The IG is a carefully planned proposal. It provides a fair basis for comparison: the woman … acts as a neutral point so that the two imposters can be assessed in how well they 'fake'." (pp. 466-7)

- But if this idea is so "carefully planned", why does the later text ignore it, and suggest that the aim is to imitate "a man"?
- The same applies to Turing's later presentation of his Test in 1952: see the underlined passages in slides 279-80 below.

237

237

"Man" = "Adult Human"
(probably, as this is common usage in 1950)

- § 2 para. 2: "The new problem has the advantage of drawing a fairly sharp line between the physical and the intellectual capacities of a man." (p. 434, *ET* 442)
- § 2 para. 2: All three example question topics are entirely gender-neutral (sonnet, arithmetic, chess).
- § 2 para. 4: "If the man were to try and pretend to be the machine …". (p. 435, *ET* 442)
- § 2 para. 5: "It might be urged that … the best strategy for the machine may … be something other than imitation of the behaviour of a man. …" (and see also slide 240 below, for more of this paragraph)

238

238

Copeland & Proudfoot (2009)

> "To assess the computer's performance, we ask:
>
> 'Will the interrogator decide wrongly as often when the [computer-imitates-human] game is played … as he does when the game is played between a man and a woman?'
>
> *[The quoted question is from Turing – slide 234]*
>
> "If the computer (in the computer-imitates-human game) does no worse than the man (in the man-imitates-woman game), it passes the test. … the man-imitates woman game is not intended as an *introduction* to the test at all, but rather as part of the protocol for scoring the test." (pp. 123-4)

239

239

§ 2 Critique of the New Problem

- § 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.
- 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". (pp. 434-5, *ET* 442-3)

240

240

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. [Note: the correct answer is 105721!]*

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.*

(pp. 434-5, *ET* 442)

241

241

★ Bias towards Human Thought?

- Turing notes the likely 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?" (p. 435, *ET* 442)
- 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.

242

242

§ 3 The Machines concerned in the Game

- "It is natural that we should wish to permit every kind of engineering technique to be used in our machines" (p. 435, *ET* 443), including experimental techniques that result in a machine whose operation is unclear even to the engineers;
- However "we wish to exclude from the machines men born in the usual manner"! ☺
- Seeking for an appropriate rule to do this, Turing restricts his test to digital computers.

243

243

★ Testing Individual Machines, or Possible Machines in General?

- Turing's texts mostly give the impression that he is advocating a test for specific machines, e.g. (in 1952) "Is *this* a Grade A machine?"
- But in § 3 of the 1950 paper, he modifies the question to discuss what an abstract digital computer could do *in principle*:

  "we are not asking whether all digital computers would do well in the game nor whether the computers at present available would do well, but whether there are imaginable computers which would do well." (p. 436, *ET* 443)

244

244

§ 4 Digital Computers

- "The idea behind digital computers [is] that these machines are intended to carry out any operations which could be done by a human computer." (p. 436, *ET* 444) – here we see clear echoes of § 9 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" (p. 438, *ET* 445).

245

245

Varieties of Digital Computer

- Turing considers some particular cases:
  - A "digital computer with a random element" (p. 438, *ET* 445), which can be simulated by including a pseudo-random process (e.g. using digits of π).
  - A "computer with an unlimited store"; "Such computers have special theoretical interest and will be called infinitive capacity computers".
- He ascribes the idea of a digital computer to Charles Babbage (who designed his "Analytical Engine" in 1837), thus illustrating that they can be mechanical and need not be electrical.

246

246

Turing on Computability and Intelligence

★ § 5 Universality of Digital Computers

- Even if processes in the world are *really* continuous (and chaotic – Turing here gives an illustration of what is now known as the "butterfly effect", p. 440, *ET* 447), some of them can be usefully modelled as discrete.
- The behaviour of "discrete state machines" is utterly predictable, and "There is no reason why this calculation should not be carried out by means of a digital computer. Provided it could be carried out sufficiently quickly the digital computer could mimic the behaviour of any discrete state machine." (p. p. 441, *ET* 448)
- In this sense, digital computers "are *universal* machines."

247

247

★ Seeking Possibility, not Actuality

- In the last paragraph of § 5, Turing refers back …

  "We may now consider again the point raised at the end of § 3. It was suggested tentatively that the question 'Can machines think?' should be replaced by 'Are there imaginable digital computers which would do well in the imitation game?' … But in view of the universality property … [this] is equivalent to …, 'Let us fix our attention on one particular digital computer *C*. Is it true that by modifying this computer …, 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?'" (p. 442, *ET* 448)

248

248

§ 6 Contrary Views on the Main Question

- Turing now sketches his own views on "our question, 'Can machines think?' and the variant of it quoted at the end of the last section [i.e. can a digital computer be programmed to pass the test?]." (p. 442, *ET* 448)
- He then considers, and rejects, nine objections: (1) "Theological", (2) "Heads in the Sand", (3) "Mathematical", (4) "The Argument from Consciousness", (5) "Arguments from Various Disabilities", (6) Lady Lovelace's Objection, (7) "Argument from Continuity in the Nervous System, (8) "The Argument from Informality of Behaviour", and (9) "The Argument from Extra-Sensory Perception".

249

249

★ Turing's Year 2000 Predictions

- But first, Turing offers his own predictions:
  1. "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." (p. 442, *ET* 449)
  2. "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." (p. 442, *ET* 449)

250

250

(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." (p. 443, *ET* 449)

- 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.

251

251

(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." (p. 444, *ET* 450)

- 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.

252

252

Turing on Computability and Intelligence

(3) The Mathematical Objection

- These objections are based on the results of "Gödel, … *Church*, *Kleene*, *Rosser*, and *Turing*" that demonstrate the limited power of "discrete-state machines" (p. 444, *ET* 450).
  - Such machines may fail on questions like those parallel to the § 8 lemma of his 1936 paper:
    "Consider the machine specified as follows ... Will this machine ever answer 'Yes' to any question?"
  - But humans also have limitations, and superiority over one particular machine is quite different from superiority over *any* machine.

253

253

(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 ..."
    (pp. 445-6, *ET* 451)

254

254

Turing's Response to Jefferson

- 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." ☺ (p. 446, *ET* 452)

255

255

- Turing now imagines a *viva-voce* (oral) examination being given to the AI composer of a Shakespearean sonnet (evidently picking up on Jefferson's reference to sonnet writing):

  "What would ... Jefferson say if the sonnet-writing machine was able to answer like this in the *viva-voce*? I do not know whether he would regard the machine as 'merely artificially signalling' these answers, but if the answers were as satisfactory and sustained as in the above passage I do not think he would describe it as 'an easy contrivance'. … In short, then, I think that most of those who support the argument from consciousness could be persuaded to abandon it rather than be forced into the solipsist position."

256

256

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.

257

257

Two Distinct Lines of Thought

- There are two quite different lines of thought in Turing's response to Jefferson, which he would have done well to distinguish:
  - Jefferson is "denying the validity" of the Turing Test *on the grounds that 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"*.

258

258

Turing on Computability and Intelligence

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.).
- I shall argue that he should instead have said, 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."

259

259

(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*. (p. 447, *ET* 453)
- The machines of 1950 couldn't do most of these, but it would require argument to show than *none* could.
- Several of these alleged limits seem just to take us back to the argument from consciousness.

260

260

Machines Can Make Mistakes

- Turing remarks on the oddity of one objection:
  - "The claim that 'machines cannot make mistakes' seems a curious one. One is tempted to retort, 'Are they any the worse for that?' ..." (p. 448, *ET* 454)
- He goes on to distinguish 'errors of functioning' from 'errors of conclusion':
  - A machine could be programmed so that it functions so as to make errors in its conclusions, precisely to imitate a human more convincingly.
  - But it still seems odd that Turing's Test might count *better* performance *against* the machine!

261

261

(6) Lady Lovelace's Objection

- Turing quotes Ada Lovelace (1842, note G):

  "[Charles Babbage's] Analytical Engine has no pretentions to *originate* anything. It can do *whatever we know how to order it* to perform" (p. 450, *ET* 455).
- In response, Turing points out that:
  - We are fallible, and can often be surprised by the novel consequences of our actions, views, and the instructions that we give to machines;
  - Claiming that we are creative in a special way leads back to the consciousness argument.

262

262

- 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

263

263

(7) Argument from Continuity in the Nervous System

"The nervous system is certainly not a discrete-state 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." (p. 451, *ET* 456)

- In response, Turing points out that a discrete-state machine could *mimic* a continuous system sufficiently closely that a human would find it extremely hard to tell the difference. This is reminiscent of his comments under § 5 above.

264

264

Turing on Computability and Intelligence

(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." (p. 452, *ET* 457)

  – 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")

265

265

(9) The Argument from Extra-Sensory Perception

- Turing's sympathy towards telepathy, clairvoyance, precognition and psycho-kinesis seems surprising: "the statistical evidence, at least for telepathy, is overwhelming" (p. 453, *ET* 458).
  – 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".

266

266

★ § 7 Learning Machines

- Turing admits at the beginning of § 7 that
  – "I have no very convincing arguments of a positive nature to support my views." (p. 454, *ET* 458)
- 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 § 6, will be that provided by waiting for the end of the century and then doing the experiment." (p. 455, *ET* 459)

267

267

- 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 imitation 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 didn't have the advantage of our further 75 years of experience, which have shown how difficult it is to implement general learning systems to compete with the products of evolution.

268

268

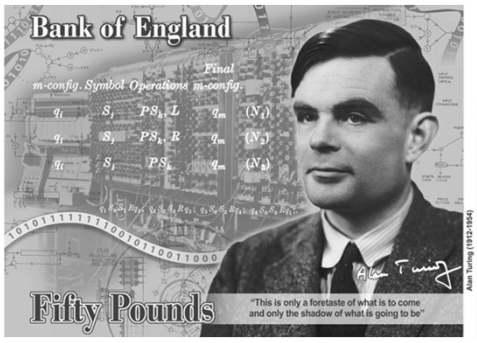
★

- Turing reiterates a point made against the Lovelace objection, that a learning machine is 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 (p. 458-9, *ET* 462-3).
- "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.
- 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." (p. 460, *ET* 463)
- On this speculative note, the paper ends.

269

269

2. The Issue of Bias



270

270

Turing on Computability and Intelligence

Two Kinds of Bias

- At first glance, Turing's Test seems obviously *biased in favour of human intelligence*: any normal adult human will easily pass, but a machine – in order to pass – has to succeed in the very difficult task of imitating a human!
- Digging deeper, though, we find that Turing's Test is specifically designed to *avoid* human-centric bias of a particular kind. This may seem initially paradoxical, but it makes sense if we focus on Turing's primary concern: *to avoid anti-machine bias by the interrogator*.

271

271

Passing as a *Sufficient* Criterion Only

- Turing acknowledged that his test might be considered objectionably human-centric:

  "May not machines carry out something which ought to be described as thinking but which is very different from what a [human person] does?" (p. 435, *ET* 442)
- His response (slide 242) seemed to suggest that he was willing to interpret passing his Test as a *sufficient* criterion of intelligence, but not a *necessary* condition. *Let us – from now on – interpret the Test in that way!*

272

272

A "Principle of Imitation"?

- Andrew Hodges, Turing's biographer, claims that he was motivated by a "principle of imitation":

  "the Turing machine [like] the 'Turing Test', was … based on a principle of imitation … modelled by considering what a human being could do when following a definite method." (Hodges 2009, p. 14).
- But we have previously seen evidence that the Turing machine was not so motivated – it seems to have been developed to address a variant of the Richard Paradox, rather than as an abstract model of human thinking. Let us likewise look into the background of the Turing Test …

273

273

From Gödelian and Halting-Style Problems to "Fair Play"

- Earlier we saw "The Mathematical Objection" from § 6 (3) of Turing's 1950 paper (slide 253) – another obvious link with his 1936 paper. This objection highlights that discrete-state machines may fail on questions of the type:

  "Consider the machine specified as follows ... Will this machine ever answer 'Yes' to any question?"
- Turing replies that humans also have limitations, and argues briefly for the sort of "fair play" which he had emphasised in a lecture of 1947 …

274

274

1947: Lecture to London Math Society (on the Automatic Computing Engine)

"with certain logical systems there can be no machine which will distinguish provable formulae … from unprovable. … Thus if a machine is made for this purpose it must in some cases fail to give an answer. On the other hand if a mathematician is confronted with such a problem he would search around and find new methods of proof … I would say that fair play must be given to the machine. Instead of it sometimes giving no answer we could arrange that it gives occasional wrong answers. …"

275

275

"But the human mathematician would likewise make blunders when trying out new techniques. It is easy for us to regard these blunders as not counting and give him another chance, but the machine would probably be allowed no mercy. In other words then, if a machine is expected to be infallible, it cannot also be intelligent. …
To continue my plea for 'fair play for the machines' when testing their I.Q. …, the machine must be allowed to have contact with human beings in order that it may adapt itself to their standards. The game of chess may perhaps be rather suitable for this purpose, …"

(*ET* 393-4)

276

276

Turing on Computability and Intelligence

1948: "Intelligent Machinery" (Report to National Physical Laboratory)

"13. Intelligence as an emotional concept

The extent to which we regard something as behaving in an intelligent manner is determined as much by our own state of mind and training as by the properties of the object under consideration. If we are able to explain and predict its behaviour or if there seems to be little underlying plan, we have little temptation to imagine intelligence. …"

277

277

"It is possible to do a little experiment on these lines, even at the present stage of knowledge. It is not difficult to devise a paper machine which will play a not very bad game of chess. Now get three men as subjects for the experiment A, B, C. A and C are to be rather poor chess players, B is the operator who works the paper machine. … Two rooms are used with some arrangement for communicating moves, and a game is played between C and either A or the paper machine. C may find it quite difficult to tell which he is playing. (This is a rather idealized form of an experiment I have actually done.)"

(*ET* 431)

278

278

1952: "Can automatic calculating machines be said to think?" (BBC)

"I would like to suggest a particular kind of *test* that one might apply to a machine. You might call it a test to see whether the machine thinks, but it would be better to avoid begging the question, and say that the machines that pass are (let's say) 'Grade A' machines. The idea of the test is that the machine has to try and pretend to be a man, by answering questions put to it, and it will only pass if the pretence is reasonably convincing. A considerable proportion of a jury, who should not be expert about machines, must be taken in by the pretence. They aren't allowed to see the machine … [but can] ask it questions, which are transmitted through to it …"

279

279

"it sends back a typewritten answer … [questions can be about] anything. … Likewise the machine would be permitted all sorts of tricks so as to appear more man-like, such as waiting a bit before giving the answer, or making spelling mistakes … We had better suppose that each jury has to judge quite a number of times, and that sometimes they really are dealing with a man and not a machine. That will prevent them saying 'It must be a machine' every time without proper consideration.

Well, that's my test. … My suggestion is just that this is the question we should discuss. It's not the same as 'Do machines think', but it seems near enough for our present purpose, and raises much the same difficulties."

(*ET* 495)

280

280

"Fair Play for the Machines"

- It seems clear that Turing's aim in this setup is *to ensure a fair test*: to eliminate potential bias by the interrogator and set up a situation in which the computer's responses can be compared fairly against those of a human. Hence the separate rooms with communication only by teletype, etc.
- But how to persuade people that a computer could potentially exhibit quite *general* intelligence (rather than something very specific, like chess)?
- Describing a futuristic robot would have been unconvincing – "just an imagined artificial person" – so *conversation* is the most plausible context.

281

281

Human Imitation: Means, Not End?

- So the role of human imitation in Turing's setup seems to be not so much to make it a *criterion* of intelligence, as to ensure a *fair test* that focuses on the relevant *behaviour* and prevents the interrogator from judging the agent by anti-machine prejudice.
- With an *unbiased* interrogator, perhaps *human-likeness* wouldn't need to be a *criterion* of success (though presumably, responding like an intelligent adult human would be sufficient to pass).
- In the next lecture, we ask how Turing's Test is best interpreted, taking various challenges into account, and how well it stands up to those challenges.

282

282

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 7

The Turing Test: Predictions, Interpretation, and Objections

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



283

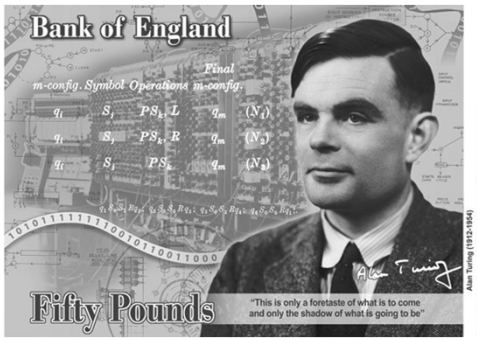
The Turing Test: Summary So Far

- In the last lecture, we reviewed Turing's 1950 paper, drawing attention to various key points, including:
  - His *Imitation Game* – now known as the *Turing Test* – seems to be presented as a criterion of machine intelligence. But he acknowledges that it is not a *necessary* criterion, for intelligence could take forms very different from the human (slide 242).
  - We saw evidence from Turing's various discussions (slides 270-82) that the role of human imitation in his setup is not so much to make it a *criterion* of intelligence, as to ensure a *fair test* that avoids anti-machine prejudice on the part of the interrogator.
  - We next turn attention towards his two 50-year predictions, for the Year 2000 (slide 250).

284

284

3. Turing's Two Predictions



285

285

Turing's First Prediction: 30% Deceit after 5 Minutes

- Just before considering "Contrary Views" in § 6 of the 1950 paper, Turing makes two predictions regarding likely progress in AI by the year 2000. The first of these has often been interpreted as proposing a *threshold* for passing the Turing Test:
  - "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." (p. 442, *ET* 449)

286

286

A Criterion of Progress Towards AI?

- Turing's way of presenting this prediction naturally suggests a *criterion of progress* towards machine intelligence, based on how well the program can fool an "average interrogator", and for how long.
- This has set a pattern for subsequent attempts to run a practical "Turing Test", with obvious ways of trying to measure increasing levels of success:
  - Extending the time;
  - Increasing the required threshold of successful deceit;
  - Widening the range of topics;
  - Involving more expert interrogators.

287

287

Mere Prediction, or a Serious Criterion?

- But did Turing really intend his "30% deceit" measure as a criterion of machine intelligence? A later back-reference (see slide 267) hints that he might have:

  "The only really satisfactory support that can be given for the view expressed at the beginning of § 6, will be that provided by waiting for the end of the century and then doing the experiment." (p. 455, *ET* 459)

- But "the view expressed at the beginning of § 6" seems to refer to Turing's two 50-year predictions, so perhaps he really does want the issue of performance on his Imitation Game to entirely *replace* the issue of whether computers can "think" or be "intelligent".

288

288

Turing on Computability and Intelligence

Turing's Second Prediction

- Turing's other 50-year prediction is that by 2000 "one will be able to speak of machines thinking without expecting to be contradicted". And indeed by 2000 this conversation was plausible:

  "Why is the computer taking so long to respond?"

  "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, I'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."

289

289

Tension Between Turing's Predictions

- Turing's first prediction concerns how successful a chatbot might be at deceiving us, while the second concerns our readiness to describe a computer as "thinking" or "intelligent".
  - In the one case, we do not know that it is a machine, but it fools us by acting in a "human-like" way.
  - In the other case, we do know that it is a machine, and its behaviour – whether human-like or not – persuades us to consider it as intelligent.
- Although Turing's focus on the Imitation Game emphasises the first approach more, there are good reasons for preferring the second …

290

290

An Unreliable Criterion

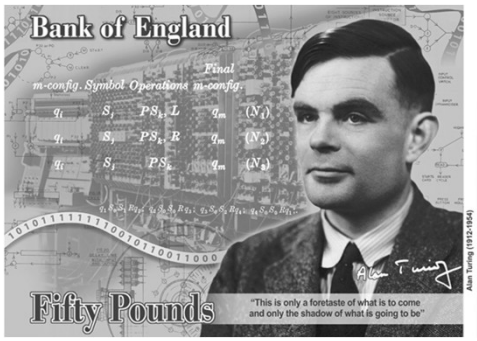
- The much-publicised threshold of *30% deceit of an average interrogator after 5 minutes* was not publicly reached by 2000. But if sustained and systematic research effort had been directed towards this goal, it probably could have been.
- But what would have made it achievable is something that Turing did not foresee, which became apparent only in 1966 …

  It turns out that fooling an average interrogator is relatively easy to achieve, but not by techniques that plausibly involve genuinely intelligent information processing.

291

291

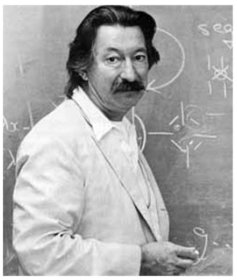
4. The Chatbot Objection



292

292

Debunking the "Turing Test": Weizenbaum's ELIZA and Chatbots



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 an event organised by the University of Reading.

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 Saint Petersburg, Russia. The development team includes Eugene's creator Vladimir Veselov, who was born in Russia and now lives in the United States, and Ukrainian born Eugene Demchenko who now lives in Russia.

The Turing Test is based on 20th century mathematician and code-breaker Turing's 1950 famous question and answer game, 'Can Machines Think?'. The experiment investigates whether people can detect if they are talking to machines or humans. The event is particularly poignant as it took place on the 60th anniversary of Turing's death, nearly six months after he was given a posthumous royal pardon.

If a computer is mistaken for a human more than 30% of the time during a series of five minute keyboard conversations it passes the test. No computer has ever achieved this, until now. Eugene managed to convince 33% of the human judges (30 judges took part - see more details below) that it was human.

293

293

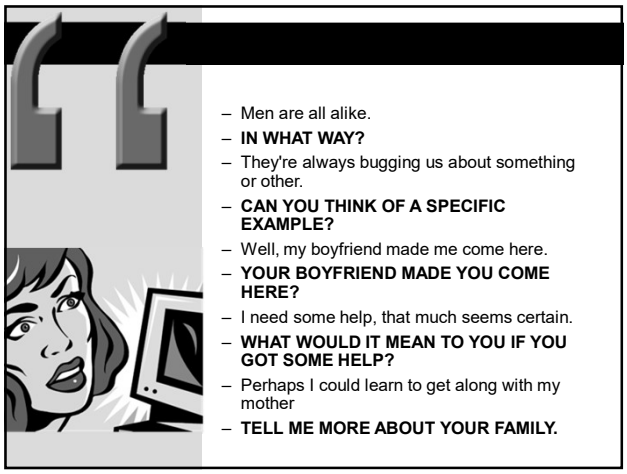
ELIZA the Psychotherapist

- In 1966, Joseph Weizenbaum published his *ELIZA* program and "DOCTOR" script, showing how simple text manipulation could generate plausible dialogue.
  - The "DOCTOR" plays the role of a Rogerian psychotherapist, largely echoing the human's input, expressing sympathy and asking gentle questions to elicit feelings.
  - Computer responses are generated by making small changes to the human inputs, exchanging first- and second-person etc.
- The original ELIZA code was unearthed in 2021: see https://sites.google.com/view/elizagen-org/the-original-eliza. For an almost-faithful and easily editable *Windows* emulation, see https://www.philocomp.net/ai/elizabeth.htm.

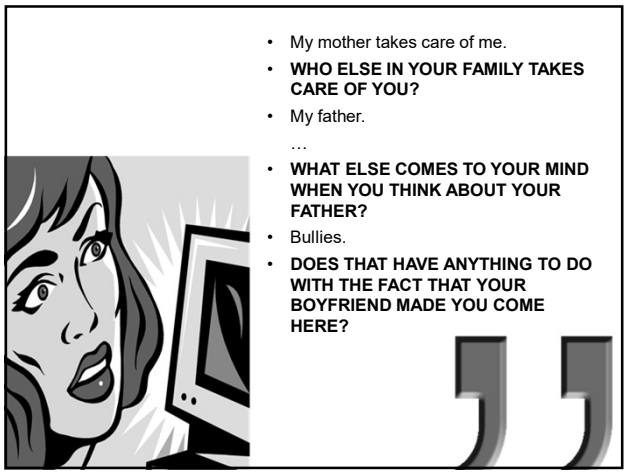
294

294

Turing on Computability and Intelligence



295



296

Simple Patterns and Responses

"… alike …"	"IN WHAT WAY?"
"something or other"	"FOR EXAMPLE?"
"My X"	"YOUR X?"
"I need Y"	"WHAT IF YOU GOT Y?"
"… my mother"	"… YOUR FAMILY …"
"My mother …"	"WHO ELSE …?"
<no matching pattern>	"DOES THAT HAVE ANYTHING TO DO WITH YOUR X?"

297

297



```
<Theme name="Mother">
 <QAUnit name="HaveMother" type="STATEMENT">
  <Pattern>* #(qDoYouHave) * #(mother) *</Pattern>
  <Pattern>* #(pYour) #(mother) * (well|feeling fine|fine) *</Pattern>
  <Pattern>mother</Pattern>
  <Random>
   <Text>Yes, of course I have mother! And I love her very much.</Text>
   <Text>My mother works on TV.</Text>
  </Random>
 </QAUnit>
<QAUnit name="MotherProfession" type="STATEMENT">
  <Pattern>* #(qWhoBeo) * #(pYour) #(mother) *</Pattern>
  <Pattern>* #(pYour) #(mother) #(qWhoBeo) *</Pattern>
  <Pattern>* (Does|if|whether|where) * #(pYour) #(mother) work* *</Pattern>
  <Pattern>* (Tell me (more|*)|) about #(pYour) #(mother) *</Pattern>
  <ContextPattern>* Who is she *</ContextPattern>
  <ContextPattern>* who she is *</ContextPattern>
  <ContextPattern>* (Does|if|whether|where) she work* *</ContextPattern>
  <ContextPattern>* what does she do *</ContextPattern>
  <ContextPattern>* #(qWhoBeo) #(pYour) #(mother) #(profession) *</ContextPattern>
<Random>
   <Text>My mother works on the Odessa TV and runs a popular program for teenagers "Speak Russian right!" Every Odessian teenager heard her famous "For those putzs who schmooze Russian in a goddam lame way: drop by and listen to mine!"</Text>
   <Text>If you lived in Odessa, you couldn't help but know my mom - she runs a popular educational TV program for teenagers about Russian language.</Text>
  </Random>
 </QAUnit>
```

"Eugene Goostman" (chatbot claimed to have "passed" the Turing Test in 2014): ELIZA-like patterns and responses

298

Making the Test More Difficult

- Any test that an ELIZA-style chatbot can pass clearly isn't a reliable test for intelligence!
  - So it's not enough just to fool "an average interrogator" 30% of the time in a 5-minute casual conversation.
- The test apparently needs to be made much more rigorous, by asking *specific*, *probing* questions that check for the ability to *understand*, in terms of *making links*, *drawing appropriate inferences* etc.
  - But the conceptual links we make in discussion often draw on all sorts of subtle features of our life and context (including human relationships). Could a computer realistically mimic these so as to be *indistinguishable*? And is such a demand sensible in a test for intelligence?

299

299

Summing Up this Discussion

- Turing acknowledges that passing his Test isn'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.
- And if interpreted rigorously, the test is *too* demanding to be useful: indistinguishability from human conversation isn't a sensible AI goal.

300

300

Turing on Computability and Intelligence

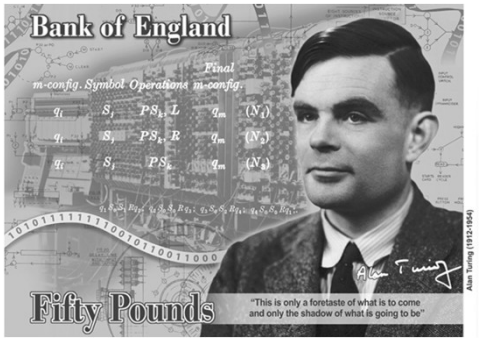
The *Tutoring* Test

- Perhaps a better form of test (suggested in Millican, 2013) 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 can it teach as well as a human?).
  - Could provide *seriously useful* products, and also stimulate progress towards "intelligent" processing.

301

301

5. A More Charitable Interpretation



302

302

Is There a Better Interpretation?

- The Tutoring Test is itself imperfect, failing to ensure "fair play" for the machine (i.e. a blind trial of the type Turing sought). But it serves a useful purpose in illustrating an advantage over the Turing Test (as usually interpreted): *we should prefer a test that reveals intelligence in action, over one that acts by concealment and deceit*.
- Is there a way of interpreting the Turing Test that avoids this flaw, as well as the other problems that we have seen? I believe there is, and it can be largely based on Turing's words, though how faithful it is to his own intentions is open to debate.

303

303

Actual or Abstract Machines?

- The most natural interpretation of the Turing Test is as a test for the intelligence of *specific* machines, e.g. "Is *this* a Grade A machine?" (slide 279).
- But as we saw, in § 3 and § 5 of the 1950 paper (slides 244 & 248), he focuses instead on what an *abstract* digital computer could do *in principle*.
- This is strongly informed by what we now call the Church-Turing thesis: that digital computers like the Turing machine are *universal* machines, capable of being adapted to perform any computable algorithm whatever.

304

304

- § 3 para. 4: "we are not asking whether all digital computers would do well in the game …, but whether there are imaginable computers which would do well."
- § 5 para. 6: "We may now consider again the point raised at the end of § 3. It was suggested tentatively that the question 'Can machines think?' should be replaced by 'Are there imaginable digital computers which would do well in the imitation game?' … But in view of the universality property … [this] is equivalent to …, 'Let us fix our attention on one particular digital computer *C*. Is it true that by modifying this computer …, 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?'"
- Now recall slide 257 – the imagined *viva-voce* examination with a sonnet-writing machine …

305

305

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.

306

306

Turing on Computability and Intelligence

The "Extreme Exemplar" Interpretation

- Turing's implicit point in presenting the sonnet discussion seems to be that *if we encountered a machine capable of responding with this degree of sophistication, in response to comparably difficult questions chosen across a wide range of subjects by some independent group of interrogators, then it would seem unreasonable to withhold the attribution of "intelligence" to that machine*.
- Hence given the universality of digital computers (which suggests that one could *in principle* be programmed to respond at this level), this "extreme exemplar" of intelligent conversation implicitly provides a sort of *existence proof of the possibility of an intelligent machine*.

307

307

Is This What Turing Intended?

- This "extreme exemplar" interpretation is a "charitable" reading of Turing's texts, which aims to be faithful to the spirit of his arguments, drawn as far as possible from his own words, and yielding overall a consistent and plausible position.
  - Whether Turing himself had this position firmly in mind when writing or saying those words is doubtful, because what he says often hints in other directions, and is sometimes vague or ambiguous. Also, his views, focus and intentions developed over time.
  - But I believe this is overall the strongest interpretation to be found, if we aspire to make maximal sense of Turing's arguments, for a maximally plausible view.

308

308

Summing Up the Turing Test

- Turing intends his Test as a *sufficient* criterion of intelligence, not a *necessary* criterion: intelligence does not require the ability to mimic a human.
- He frames his criterion in the form of an "imitation game" primarily to make the test "blind", in order to screen out *interrogator bias* against AI.
- But if the threshold of imitation is set too low (e.g. 30% successful deceit after 5 minutes with an average non-expert interrogator), then the Test is refuted by ELIZA-style chatbots, whose crude operation clearly cannot count as a plausible demonstration of genuine "intelligence".

309

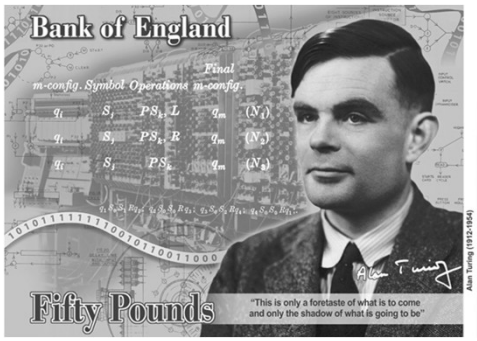
309

- But if the threshold is set high enough to exclude such a chatbot – so that passing requires the ability to process complex conversation (as, exhibited, for example, by Turing's sonnet discussion) – then it is very hard to avoid pro-human bias, because our conversation so often relies on, and displays, subtle features of culture and context.
- The "extreme exemplar" interpretation aims to avoid this sort of practical problem, by treating the Test as "propaganda" (as in slide 228): a *thought-experiment* to persuade us of the *plausibility* of AI, based on the *possibility* of unambiguously intelligent linguistic behaviour. (For more on this, see my paper of 2021.)
- But is linguistic *behaviour* – even at the level of the sonnet discussion – enough to prove intelligence?

310

310

6. Blockhead



311

311

Ned Block's "BlockHead"

- In 1981, Ned Block attacked the Turing Test on the grounds that it could in principle be passed by an obviously *mindless* machine.
  - His postulated machine, widely known as "Blockhead", works by storing every possible "sensible" sequence of inputs of a given length, together with pre-written responses that are appropriate in each case, in a giant "look-up table".
  - Something working so mindlessly, by a mechanism that involves no understanding, surely cannot be called *intelligent*, but it will always generate sensible conversation!

312

312

Turing on Computability and Intelligence

A Reality Check

- How many "sensible" conversations are there, involving, say, 10 short sentences (average 10 words) from each of the two participants?
- Suppose on average each word is chosen from a limited menu of 100 choices.
  - Number of conversations ≈ $100^{200} = 10^{400}$.
  - The known universe contains around 10^{80} atoms. Replace each atom with a complete universe (making 10^{160} atoms). Repeat three more times (10^{240}, 10^{320}, 10^{400}) – that gets you to this number of conversations (and these are only *short* ones)!

313

313

Definitive Criterion, or Evidence?

- 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.).

314

314

Intelligence and Resources

- Our "intuitive" judgements in response to Blockhead (etc.) suggest that our notion of *intelligence* involves not just impressive behaviour, but impressive behaviour *generated "cleverly" with limited resources* (e.g. of storage, information, and time).
  - Note that *in practice, the two come to the same thing*. Thought-experiments can float free of any plausible reality, but *in practice*, appropriately flexible and timely responses to a rapidly changing situation are achievable only by clever processing.

315

315

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

316

316

- The classical intuition pumps to which Dennett refers are designed to illuminate one thing – say the nature of the world, or the social order – by harnessing our *familiar* understanding of something else – shadows cast by a fire in a cave, or contracts between individuals.
  - We are thus encouraged to see the one thing as relevantly similar to the other.
  - Computer analogies can also work in this way, e.g. comparing the mind to a computer program, or religion to a computer virus (Dawkins).
- But different analogies can suggest quite different conclusions, as becomes clear if we compare Turing's position with that of Block.

317

317

Block *versus* Turing

- We'll see more of this later (when we come to Searle), but for now, notice the intimate contrariety between Block's hypothetical "Blockhead" program 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 and appealing to contrary "intuitions" …

318

318

Turing on Computability and Intelligence

- Block: "Imagine a computer program that conducts its conversation using a crude lookup table like this …
  - ... how could you possibly claim that it is genuinely intelligent?"
- Turing: "Imagine a computer program that is able to converse like this, so as to pass convincingly for a human …
  - ... how could you possibly deny that it is genuinely intelligent?"
- I suggest (controversially) that the contest between these rival points of view should be adjudicated by appeal to *practical plausibility*.

319

319

- 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 grand-master 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.

320

320

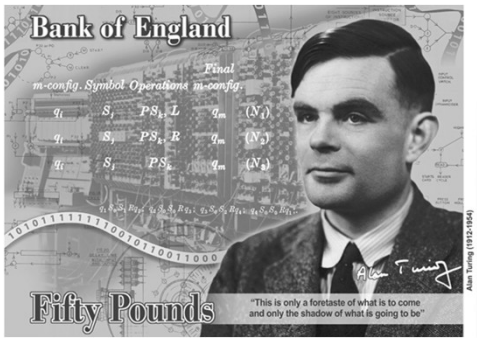
A More Fundamental Objection

- We must now consider an even more fundamental objection to the Turing Test, which claims that there is a theoretical error in supposing that a conversational AI system – however excellent its performance – could properly be considered "intelligent", on the grounds that its internal processing fails to achieve any genuine *understanding* or *semantic grasp* of the topics that it purports to be discussing. This is John Searle's *Chinese Room* thought-experiment.

321

321

7. Searle's Chinese Room



322

322

Searle's "Chinese Room"

- John Searle's famous thought-experiment involves a conversation conducted in written Chinese by means of cards posted into and out of a room with a single occupant:
  - incoming cards express meaningful questions, written by a competent Chinese speaker;
  - outgoing cards – written by the man within the room (e.g. Searle himself) – provide meaningful and appropriate answers to those questions, such as might be produced by a competent and intelligent native speaker of Chinese.

323

323

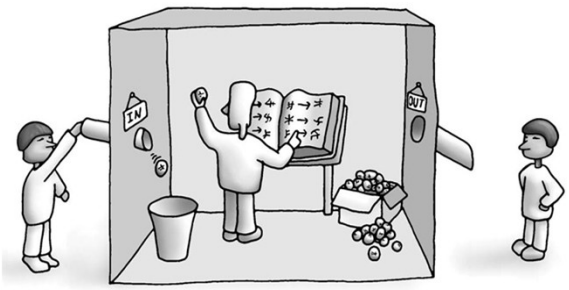
- But the man inside the room has no knowledge whatever of the Chinese language or of the *semantics* – the meaning or significance – of the symbols he is reading or writing.
- Instead, he is generating his written "answers" by strictly applying rules based purely on the *syntax* – the shape and structure – 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)

324

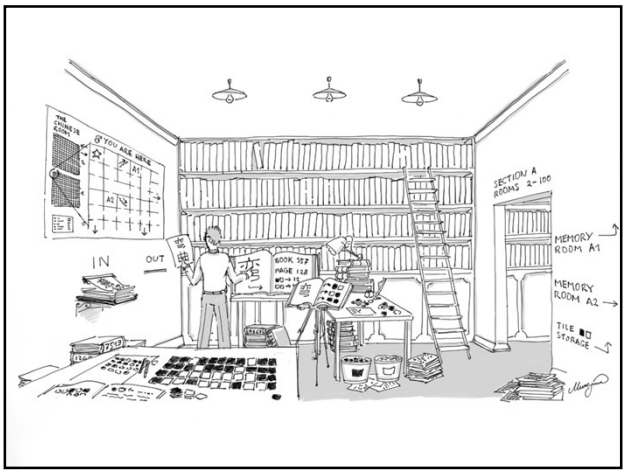
324

Turing on Computability and Intelligence

This depiction of the scenario (from *Wikimedia Commons*) is hopelessly unrealistic, like a grandmaster chess program of only 50 lines. Vladimir Mikulik has kindly created an image that far better captures the complexity of the situation …



325



326

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.
- 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!).

327

327

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)

328

328

What Exactly is Searle Denying?

- Most of the time, Searle expresses his thesis as a denial of "intentionality" or "semantic content".
- 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.)

329

329

The Cautious Interpretation

- Searle is often interpreted as arguing that "machines cannot think", but he accepts that <u>we</u> are thinking "machines" (in the sense of physical things working according to natural laws).
- When most cautious, he seems to be objecting to the idea that *<u>digital computers</u> can have semantic (i.e. meaningful, genuinely intentional) states <u>purely in virtue of following a symbolic algorithm</u>.*
  - But he often seems to go significantly beyond this cautious interpretation (e.g. when addressing the "robot reply", which we consider shortly).

330

330

Turing on Computability and Intelligence

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)

331

331

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.

332

332

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 …

333

333

"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)

334

334

An "Intelligent" Tutoring System?

- Imagine (as previously) a computerised tutoring system which is capable of highly sophisticated information processing in a complex domain.
  - 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" or "intentionality".
  - It has no real *understanding* of the domain – its "thought" cannot "reach out" to the real world.

335

335

Intrinsic/As-If/Derived Intentionality

- Searle's *The Rediscovery of the Mind* (1992, pp. 78-9) distinguishes:
  - *Intrinsic intentionality* involves "a real, intentional, mental state", e.g. when I say "I am now thirsty".
  - *As-if intentionality* is metaphorical, for example if I say "My lawn is thirsty". The lawn has no genuine intentional states, but I am talking of it as-if-it-had-intentionality.
  - "In French, 'j'ai grand soif' means 'I am very thirsty" illustrates *derived intentionality*: derived from the intrinsic intentionality of language users.

336

336

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 8

Intelligence, Intentionality, Consciousness, and Changing Paradigms



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

337

Summary on Searle So Far

- Searle's Chinese Room thought-experiment aims to convince us (arguably an an "intuition pump") that mere *syntactic manipulation* of symbols cannot invest them with *semantic content* or genuine *intentionality*. Hence contrary to Turing, "strong AI" is impossible: digital computers cannot *think*.
- We have briefly noted two main objections, the "System Reply" and the "Robot Reply":
  - Searle's answer to the System Reply seems to beg the question, as a matter of logic, but retains some intuitive appeal. We have yet to assess his dismissive answer to the Robot Reply.

338

338

Possible Responses to Searle: 1. Intelligence without Intentionality

- We can agree with Searle that our imagined tutoring system has no *intrinsic, genuinely-system-understood, intentionality*. It does not "understand" that it is processing information about (say) chemistry.
  - We might nevertheless insist that its *processing* of information is "intelligent", on the basis of its impressive performance (in line with the "extreme exemplar" interpretation of the Turing Test).
  - After all, we seem to be able to judge such a system's performance as *more*, or *less*, intelligent, without worrying about the status of its semantics.

339

339

Large Language Models

- Deep-learning on massive language corpora has recently delivered very surprising results, whereby systems such a *ChatGPT* appear to exhibit mastery of complex conceptual links.
  - This suggests that such systems could *learn* the sort of "quasi-understanding" that the postulated Tutoring System would exhibit, *all at once, and across a huge domain*.
- Such competence might seem to be exhibited by *ChatGPT*'s response to Turing's question "Please write me a sonnet on the subject of the Forth Bridge" (p. 434, *E* 442, slide 241):

340

340

Amidst the Scottish landscape, strong and grand,
Stands a marvel, a sight for all to see,
A testament to skill of human hand,
The mighty Forth Bridge, spanning wide and free.
Its iron girders reach towards the sky,
A symphony of strength and engineering,
Connecting lands where mighty rivers lie,
Defying nature's whims with persevering.
For generations, it has proudly stood,
A symbol of resilience and resolve,
Its arches and supports, a steadfast brood,
Withstanding time, as countless years evolve.
Oh, Forth Bridge, your beauty knows no bound,
A tribute to the marvels man has found.

341

341

Back to the Turing Test?

- Note that the "extreme exemplar" interpretation *does not* take us back to the (chatbot-refuted) idea that "intelligence" is displayed by a Large Language Model through doing "well" in a Turing Test of the traditional kind, e.g. by faking humanity over a chat of 2 or 5 minutes respectively …
  - https://arxiv.org/abs/2305.20010: Daniel Jannai, Amos Meron, Barak Lenz, Yoav Levine, Yoav Shoham (2023), "Human or Not? A Gamified Approach to the Turing Test"
  - https://arxiv.org/abs/2405.08007: Cameron R. Jones and Benjamin K. Bergen (2024), "People cannot distinguish GPT-4 from a human in a Turing test"

342

342

Turing on Computability and Intelligence

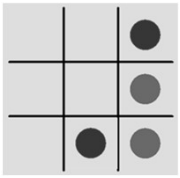
2. Achieving "Intrinsic Intentionality" in Abstract Domains?

- Searle's argument seems most convincing when he denies that syntactic processing can generate "intrinsic intentionality" towards physical things (e.g. chemicals, trees, bridges).
- But a similar denial is less convincing when considering "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 or bridges, but only to logical relations, which can apparently be represented formally.

343

343

A Simple Example



- Suppose we have a program playing noughts-and-crosses by calculating the best move in every situation (analysing all possible moves and recursively *minimaxing* – this logic is explained in slides 43 to 48 below).
- Obviously the program isn't *aware*, but its processing is *appropriately responsive* to the real logic of the game (e.g. how one position changes to another, how positional assessments are inherited through the game analysis tree). Shouldn't that be enough for "intelligence"?

344

344

From Abstract to Concrete?

- If this much is accepted, suppose we now choose a *concrete* domain whose behaviour can be modelled accurately – and usefully – in terms of abstract rules (e.g. a physical system such as planetary motion, a biological system such as cellular or evolutionary mechanisms, or an artificial system such as trains on a network).
- Again, if our computer system is appropriately responsive to the real "logical structure" of the domain – even if that reflects contingent properties rather than abstract relations – it's not *obviously* false to claim that the system *genuinely* represents the domain's structural relationships.

345

345

Modelling or Mimickry?

- It is unclear whether this sort of argument (even if accepted) could plausibly be extended to a machine-learning system such as a *Large Language Model* …
- Here, our judgement might depend on:
  - how far we consider their (potentially impressive) behaviour to be a reflection of their having acquired *a faithful internal model* of the relevant domain;
  - how far we think they are simply behaving as a "stochastic parrot", by simply mimicking – with random variations – the textual patterns from their training data.
- Whether an LLM can generate a useful internal model from training data may depend on the domain, and is currently controversial in many areas!

346

346

3. Going "Concrete" Through Robotic Interaction?

- If, on the other hand, we are convinced that real, physical, interaction is required in order to achieve genuine "intentionality", then we can fall back on what Searle calls the "Robot Reply".
- The idea of this is that a robotic AI can *literally* "reach out" semantically to things in the world by detecting them (e.g. with cameras) and affecting them (e.g. with robotic limbs or tools).
- If its symbolic manipulations are thus responsive to real things, and interact with them accordingly, doesn't that achieve *intrinsic intentionality*?

347

347

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

[* a Monty Python reference]

- Imagine, for example, a robotic crane, armed with appropriate sensors and tools, 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 processing seems to achieve real intentionality: its internal states are responsive to physical things, and impact causally on them.

348

348

Turing on Computability and Intelligence

Searle's Robot Response Again

- Searle will still deny that the robotic crane's internal states have "semantic content":
  - His response to the Robot Reply suggests we consider a "Chinese cabin" in the crane, fed with messages from the sensors written in Chinese characters, processed purely syntactically by the man inside, and resulting in Chinese messages to the crane's motors – all of these messages being incomprehensible to him.
  - But this response is only conclusive if Searle's rebuttal to the System Reply works …

349

349

Combining Robot and System Replies

- Let us allow, with 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.

350

350

What Do Machines Lack?

- Searle will 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-Room-style arguments suggest he thinks "understanding" in the sense of *conscious awareness* is the crucial factor.
  - (Note in passing that "understanding" is a slippery word, and often context-relative. So it's problematic as a philosophical requirement, without first clarifying what it's intended to involve, and why.)

351

351

Animals Have What It Takes

- Searle's discussion of animals confirms his focus on subjective awareness:

> "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

352

352

A Nuance: *Potentially* Conscious

- Elsewhere, Searle is more explicit on the relation between intentionality and consciousness, while also recognising (what might otherwise seem a serious objection) that we can have *unconscious* "intentional states":

> "Only a being that could have conscious intentional states could have intentional states at all, and every unconscious intentional state is at least potentially conscious."
>
> *Rediscovery of the Mind* (1992), p. 132

353

353

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.

354

354

Turing on Computability and Intelligence

How to Oppose Searle

- The simplest way of opposing Searle is to draw a clear distinction between:
  - *Information processing* and its characteristics.
  - The extent to which that processing is *conscious.*
- The former category could be divided further:
  - Information processing which is embodied in such a way as to have direct "quasi-semantic" connections with the relevant subject-matter;
  - Information processing which requires interpretation by some external agent.

355

355

Three Categories of "Intelligence"?

- Categorised like this, information processing (about "concrete" external objects) can be:
  - Conscious and embodied
    - Humans, animals
  - Unconscious and embodied
    - The robotic lumberjack
  - Unconscious and second-hand
    - Human use of "artificial intelligence" systems
- Is there any good reason for restricting words such as "intelligence" to the first of these?

356

356

Embracing Conceptual Change

- Most discussions of the Turing/Searle and related debates apparently proceed on the assumption that we should be seeking a verdict (supporting one side or the other) based on our *current* conceptual repertoire. But science does not proceed like this, and nor – I suggest – should philosophy, especially when our philosophical conundrums are prompted by *new realities* that challenge traditional boundaries.

(For more on all this, see my paper, "The Philosophical Significance of the Turing Machine and the Turing Test", 2013, from https://www.millican.org/research.htm)

357

357

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." (slides 250, 289)

- 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.

358

358

Real Novelty Beats Imagined Similarity

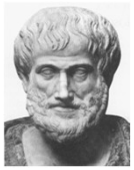
- Speculative "intuition pumps" (like Blockhead or the Chinese Room) can easily tempt us into *misleading* and *simplistic* comparisons, based on uncritical "intuitive" assumptions about supposed possibilities that are often wholly detached from reality.
- *Practical confrontation with new realities* can be far more vivid and surprising, forcing us to seriously revise our thinking by making it impossible to ignore *real possibilities* that are *unintuitive*.
- Major scientific progress has often involved such major novelties, bringing "paradigm shifts" and fundamentally new *methods of explanation*.

359

359

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.



360

360

Turing on Computability and Intelligence

The Scientific Revolution (1609-87)

- 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.



361

361

The Naturalistic Turn (1739-1859)

- Naturalist Psychology (Hume)
  - Human behaviour is governed more by emotions, imagination, and non-rational mechanisms than it is by reason: it is more "animal-like" than "god-like".



- Evolution (Darwin, Wallace)
  - Biological organisms take the form they do owing to inheritance of characteristics and competition for survival and reproduction.

362

362

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 motion.
- 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".

363

363

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.

364

364

Darwin, Einstein, and Hume





- Darwin read Hume's *Enquiries*, *Essays*, *Natural History of Religion*, *History*, *Dialogues*, and biography. Indeed he was reading and noting Hume on "the reason of animals" when he came up with his theory of evolution (1838-9).
- Einstein said that "the critical reasoning required … [to question] … the absolute character of time … was decisively furthered … by David Hume's writings". "I studied [his] treatise on understanding … with eagerness and admiration shortly before finding relativity theory."

365

365

The Weirdness of Physics

- 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 computing an answer.
  - About the nature of "thinking" itself: we need to be open to the possibility of inanimate thought.

366

366

Turing on Computability and Intelligence

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 processing 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*.

367

367

Naïve Views about Intelligence

- 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?

368

368

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 perp-etuity … *which* ship has them? (Copeland, pp. 52-3)

369

369



"Station Master says, Mum, as Cats is 'Dogs', and Rabbits is 'Dogs', and so's Parrots; but this 'ere Tortis' is a Insect, so there ain't no charge for it!"

Punch, 6 March 1869, p. 96

370

370

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.
- So when revising our concepts, we have every right to ignore crazy thought-experiments!

371

371

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.

372

372

Turing on Computability and Intelligence

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).

373

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.
- 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 problematic)!

374

Turing's Solipsistic Mistake

- As we saw (slides 254-9), Turing himself fails to distinguish the question of information processing from phenomenology, and resorts to a comparison with solipsism, which is (intentionally) amusing, but if taken seriously, would be rather silly!
- Denying that our friends and family are "phenomenally conscious", in much the same way as we are ourselves, would rightly seem ridiculous.
- 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.

375

375

An Argument Against Computers Being "Phenomenally Conscious"

- If Searle had been content to argue that digital computers cannot be *phenomenally conscious* (rather than that they cannot have *intentional states*), then he could have appealed to an argument which seems much stronger.
- This new argument is based on the claim that phenomenally conscious states (e.g. feelings and pains) are *concrete* rather than *abstract*. They are *caused* by our brains' activity (in ways that we do not currently understand), and *causally efficacious* in bringing about our actions.

376

376

Consciousness is not Abstract

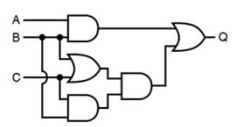
- *Intelligent information processing* (e.g. in calculation of chess moves) seems to be an *abstract* quality, potentially instantiated in the algorithmic operation of digital systems without requiring any concrete reality beyond what is needed to represent and operate on the relevant information. *Phenomenal consciousness* seems to involve some additional concrete reality.
- If this is right, then *First*, we know (in a sense to be explained) the complete explanation for the computer or robot's "intelligent" behaviour, and it involves purely abstract processes, following a program which is designed to implement sophisticated mathematics etc.; *nothing whatever to do with consciousness*.

377

377

What Ultimately Causes Computers to Work as They Do?

- Computer programs are standardly written in a *high-level programming language* (e.g. Python), but when they run, this is translated – or *compiled* – into low-level *machine code*.
- The machine code instructions are themselves *decoded* by the computer's central processing unit, and ultimately generate signals that activate circuits of *logic gates* (a simple example is shown here).



378

378

Turing on Computability and Intelligence

Complete Understanding (in a sense)

- In a sense, we have *complete understanding* of how these processes work. The processes of "translation" from high-level language, through machine code, all the way down to logic gates, are precisely specified and artificially created.
  - The *physics* of the logic gates might be mysterious at the lowest level (e.g. quantum mechanics and materials). But the *abstract behaviour* of the logic gates can be modelled mathematically, and they are engineered (e.g. with redundancy and error correction) so that the physics is not allowed to "interfere" with this abstract behaviour.
  - So in the calculation of the computer's high-level output, there is no room for *mysterious causal factors*.

379

379

- Even if the *detailed operation* of a complex program is largely mysterious to us (e.g. a Large Language Model whose capacities surprise us), we know that it is operating using standard components (e.g. silicon chips etc.) and standard processes (e.g. computer programs). We have *no reason whatever* to suppose that some mysterious conscious process is playing a causal role, and *every reason* for denying this (since we have – in general terms – that complete understanding of how the system is operating).
- *Secondly*, We have excellent reason 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 currently have almost no understanding of how it operates!*

380

380

Consciousness and Evolution

- We can be confident that consciousness must has some evolutionary function.
- Common experience tell us that it strongly influences our behaviour, and consciousness seems to be finely tuned to track things that have an impact on our evolutionary success:
  - Conscious pleasures are strongly correlated with things that have been evolutionarily beneficial (e.g. physical health, calories, social intimacy).
  - Conscious pains are strongly correlated with things that have been evolutionarily harmful (e.g. injury, starvation, social ostracism).

381

381

Consciousness is a Causal Process

- 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!*
- 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 as we have seen, there is *no such reason* to attribute it to computers, and compelling reason not to.

382

382

Summary of the New Argument

- Consciousness affects our behaviour – it is a causally active "concrete" process.
  - This is both evident, and evolutionarily required.
- Although the behaviour of computers can be very complicated, their fundamental causal processes are extremely well understood, and abstract away from any physical "mysteries".
  - So there is no room for some *other* causally active process to make a difference to how they behave.
- We do not understand our consciousness, but we do understand enough about how computers work to rule out their being conscious.

383

383

"Computing Machinery and Intelligence"

1. The Turing Test provides a context in which a machine's intelligence can be judged without bias, on the basis of performance alone: "fair play for the machines".
2. Under the "extreme exemplar" interpretation, the aim is to make a point about the *potential performance* of imagined machines, rather than actual performance.
3. It is an interesting and debatable question whether, thus understood, Large Language Models have passed the threshold of deserving the accolade of "intelligence".
4. This issue is best separated from that of consciousness, accepting that our concepts might need to change in the light of new realities. We can (and perhaps should) accept the possibility of genuine machine intelligence, without being drawn into fantasies about conscious machines.

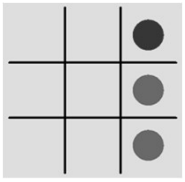
384

384

Turing on Computability and Intelligence

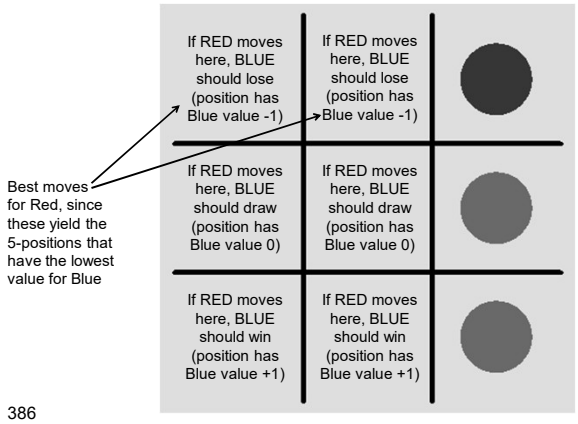
Appendix: Noughts-And-Crosses

- Playing infallibly requires a reliable method for assessing the "value" of any position (+1=winning; 0=drawing; -1=losing).
- Here it's Red's turn to move:
  - It's a "6-space-position" (let's say "6-position" for short).
  - After Red makes her move, Blue will be faced with a 5-position in which to make his reply.
  - 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?



385

385



386

386

"Minimaxing"

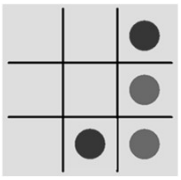
- Red winning = Blue losing (and vice-versa).
  - So the value of a position to Red is the inverse of its value to Blue (e.g. -1 as opposed to +1).
  - Hence Red tries to find moves which *maximise* the value to her, and *minimise* the value to Blue.
  - Red, faced with a 6-position, chooses the move 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) …

387

387

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.



388

388

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!
- The only other thing we need is to be able to recognise a FINAL position as winning for Red, winning for Blue, or drawn.

389

389

- 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) …
    - Evaluate each of these new positions *from Y's point of view* (resulting in evaluations Y_1, Y_2, and 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.

390

390