Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Computer Science and Philosophy, TT 2018



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 and Philosophy.*
  - The 3-hour examination will contain six "either … or" questions on "General Philosophy" and six questions on Turing.
  - Students must answer four questions, including at least one from each half of the paper (so between one and three on Turing).

2

2

Overall Plan of the Lectures

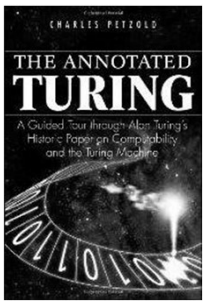
- Lecture 1: Types of number, Cantor, infinities, diagonal arguments
- Lecture 2: Hilbert's Programme, Gödel's Theorem
- Lectures 3-5: The 1936 paper, "On Computable Numbers, with an Application to the *Entscheidungsproblem*", which introduced the Turing Machine.
- Lectures 6-8: The 1950 paper, "Computing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

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

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

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 Tenth Problem

- Hilbert's list later grew to 23, including:

  10. Determination of the Solvability of a Diophantine Equation

  Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers*.

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

9

9

Diophantine Equations

- These are named after problems discussed by the 3rd century Alexandrian mathematician Diophantus, often called "the father of algebra" though some of his methods were actually Babylonian.
- Diophantine equations are polynomial equations (which may have many, or no solutions) whose solutions *must be integers*. The most famous are associated with Fermat's "Last Theorem":

  $x^n + y^n = z^n$

  Fermat claimed to have a proof that there are no integer solutions if $n>2$. This "theorem" was finally proved by Andrew Wiles (Merton College) in 1995.

10

10

Some of Hilbert's Other Problems

- 1. The continuum hypothesis: $\aleph_1 = 2^{\aleph_0}$?

  This asks whether the second transfinite number is the cardinality of the continuum – see Petzold p. 32. It has since been shown to be neither provable nor disprovable within standard set theory.
- 2. Prove the consistency of arithmetic axioms;

  See Petzold p. 46, and discussion below.
- 8. Riemann hypothesis and other problems including Goldbach's Conjecture;

  Goldbach is mentioned by Petzold on p. 47.

11

11

Hilbert's Formalist Challenges

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

12

12

Turing on Computability and Intelligence

Turing's 1936 Paper:

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

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

13

13

Various Types of Number

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

14

14

Various Types of Infinity



- Georg Cantor (1845-1918) developed set theory and the theory of *transfinite cardinal numbers*.
- His work implies that we must recognise more than one "infinite number".
- The key to understanding such numbers is widely known as "Hume's Principle" …

15

15

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

16

16

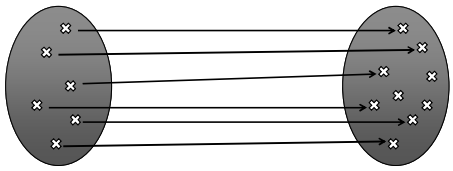
One-to-one Correspondence

- 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).

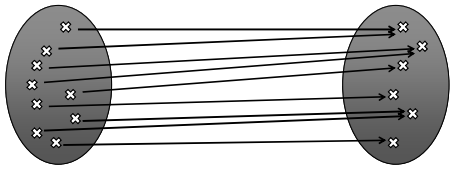
17

17

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



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



18

18

Turing on Computability and Intelligence

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| ≥ |B|. (Intuitively, A must provide enough "arrows" to hit all of B.)
- Surjections in both directions would imply

  (|A| ≥ |B|) ∧ (|B| ≥ |A|) ∴ |A| = |B|

19

19

Enumerability / Countability

- If a surjection can be defined from the set of natural numbers ℕ to set A, then we say that A is *countable* or *enumerable* (i.e. they can in principle be *enumerated*, or put in a list).
- Here, following Petzold (and tradition), we take the natural numbers to start from 1 (rather than from 0, as many authors do).
- Georg Cantor proved that the set of rational numbers ℚ and the set of algebraic numbers are both enumerable.

20

20

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:

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

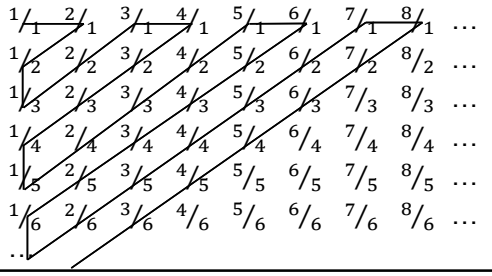
  $= \frac{367}{\cancel{1000}} \times \frac{\cancel{1000}}{999}$ [using the formula "$a/(1-r)$"]

21

21

Rational Numbers are Enumerable

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



22

22

- 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).

23

23

√2 is Not Rational

- 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!!

24

24

Turing on Computability and Intelligence

Algebraic Numbers

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

25

25

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.

26

26

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

27

27

- 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.

28

28

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*.

29

29

Real Numbers are Not 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: $0.a_{11}a_{12}a_{13}a_{14}a_{15}a_{16}a_{17}a_{18}\ldots$
  a_2: $0.a_{21}a_{22}a_{23}a_{24}a_{25}a_{26}a_{27}a_{28}\ldots$
  a_3: $0.a_{31}a_{32}a_{33}a_{34}a_{35}a_{36}a_{37}a_{38}\ldots$
  a_4: $0.a_{41}a_{42}a_{43}a_{44}a_{45}a_{46}a_{47}a_{48}\ldots$
  a_5: $0.a_{51}a_{52}a_{53}a_{54}a_{55}a_{56}a_{57}a_{58}\ldots$
  … …

30

30

Turing on Computability and Intelligence

Diagonalisation

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

  If a_{nn}=5, then c_n=0, otherwise c_n=5.

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

31

31

- 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.

32

32

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

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

33

33

An Infinity of Infinities

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

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

- S cannot be in the range of F, because if S=$F(x)$ for any x, then $x \in S \leftrightarrow x \notin S$.
- This implies that the power set of any infinite set must have a cardinality strictly greater than that set – we cannot stop at $2^{\aleph_0}$ (which is the cardinality of the continuum – Petzold pp. 31-2).

34

34

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.

35

35

- 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."

36

36

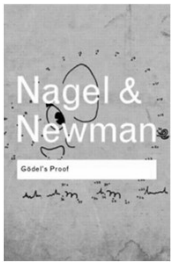
Turing on Computability and Intelligence

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

37

Hilbert's Programme (from lecture 1, slide 12)



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

38

38

Consistency, Completeness, and Decidability

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

39

39

Syntactic Formalism

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

40

40

Consistency and Completeness

- Thus *consistency* (cannot prove both *P* and ¬*P*) and *completeness* (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 consistent and complete system of arithmetic, with *true* axioms and *valid* rules, then any arithmetical proposition would be provable if, and only if, it is true. A major part of Hilbert's dream would thus be realised.

41

41

Gödel's Theorem (1931)



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

42

42

Turing on Computability and Intelligence

The Proof Strategy

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

43

43

Gödel Numbering

- Gödel's proof (*for which I mainly follow Nagel and Newman's presentation*) encodes statements *about* mathematical relationships (e.g. that some sequence of formulae provides a valid proof of some formula *F*) as formulae *within* arithmetic.
  - This involves assigning a *Gödel number,* **g[*f*]** for short, to each formula *f*, according to its structure:
    - Constant symbols have **g** numbers 1 ("¬"), 2 ("∨"), 3 ("→"), 4 ("∃"), 5 ("="), 6 ("0"), 7 ("s"), 8 ("("), 9 (")"), and 10 (",").
    - Numerical variables have **g** numbers 11 ("*x*"), 13 ("*y*"), 17, 19, …
    - Sentential variables have **g** numbers 11^2 ("*p*"), 13^2 ("*q*"), 17^2, …
    - Predicate variables have **g** numbers 11^3 ("*P*"), 13^3 ("*Q*"), 17^3, …

(11, 13, 17, 19, … is following the sequence of prime numbers >10)

44

44

- Having Gödel numbers for the individual symbols, a number can now be worked out for each *formula* by raising successive prime numbers to those powers:

$\exists$	x	(	x	=	s	y	)
4	11	8	11	5	7	13	9

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

- Then a Gödel number for a *sequence of formulae* can be derived similarly, by raising 2 to the power of the first formula, 3 to the power of the second, and so on, then multiplying these all together. Obviously, the numbers are astronomical!

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

45

45

Decoding Gödel Numbers

- Note that each formula (or sequence of formulae) has a *unique* Gödel number – no two formulae (or sequences) can have the same Gödel number.
- Moreover it is straightforward "in principle" to decode a Gödel number, by factorising it into its prime factors, examining the powers of those factors, etc.
- This makes it feasible to use Gödel numbers as proxies for those formulae in expressing their properties and the relations between them (e.g. the function we shall call *sub* that holds when one formula is a substitution instance of another).

46

46

Substituting Numerals for Variables

- We earlier saw how to derive the Gödel number of the formula:

  $\exists x\ (x = sy)$

  $2^4 \times 3^{11} \times 5^8 \times 7^{11} \times 11^5 \times 13^7 \times 17^{13} \times 19^9$
- 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 formula is:

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

47

47

- So here we have taken formula number:

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

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

  and within it we have substituted the variable with number:

  13 "*y*"

  by the numerical expression for the number:

  2 "ss0"
- This substitution has yielded formula number:

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

  "$\exists x\ (x = sss0)$"

48

48

Turing on Computability and Intelligence

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

  $2^4\times3^{11}\times5^8\times7^{11}\times11^5\times13^7\times17^7\times19^7\times23^6\times29^9$
  $= sub(2^4\times3^{11}\times5^8\times7^{11}\times11^5\times13^7\times17^{13}\times19^9, 13, 2)$

- 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. 13 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.

49

49

- *sub* is arithmetically a complicated function, but it is well-defined and could be spelled out in detail.
- If F_y is a formula containing y (Gödel number 13) and F_n is the corresponding formula in which y is substituted by the numeral for n, then we have:

  $\mathbf{g}[F_n] = sub(\mathbf{g}[F_y], 13, n)$

- Imagine now that the function $sub(a, b, c)$ is spelled out in gory arithmetical detail, and that we write out this expression with y in place of a and c, and 13 in place of b. This yields a complicated (but provably possible) arithmetical expression for:

  $sub(y, 13, y)$

  which will soon play a key role.

50

50

Arithmetising Meta-Mathematics

- *sub* is a *relatively* simple function. But Gödel also showed that it is possible to define a (far more complicated) *arithmetical* formula *A* corresponding to the *meta-mathematical* statement that a sequence of formulae *S* constitutes a proof of formula *F*. *A* will be (arithmetically) true if and only if *S* indeed proves *F*.
- Let us use "Proof(a, b)" as shorthand for the *arithmetical* formula corresponding to:

  "The sequence of formulae with Gödel number a is a proof (or demonstration)* of the formula with Gödel number b."

51

** Nagel and Newman use "Dem" instead of "Proof"*

51

A Truth-Preserving Correspondence

- The crucial point here is that the *arithmetical formula* Proof(a, b) will be true if, and only if, the *meta-mathematical statement* "The sequence of formulae with Gödel number a is a proof of the formula with Gödel number b" is also true. *[Note that Proof(a, b) expresses a statement – e.g. of an arithmetical equation – rather than a mere numerical expression like sub(a, b, c)]*
- So to establish whether or not the sequence of formulae with Gödel number a is in fact a valid proof of the formula with Gödel number c, it suffices to establish whether or not the numbers a and c yield a true equation when substituted to give the arithmetical formula Proof(a, c).

52

52

The Heart of Gödel's Argument

- Consider now the arithmetical formula:

  $\neg$ Proof(a, b)

- This corresponds to the statement that the sequence of formulae with Gödel number a *is not* a proof of the formula with Gödel number b.
- Likewise this arithmetical formula: *

  $\neg\ \exists x$ Proof(x, *sub*(y, 13, y))

  Corresponds to the statement that there is no proof of the formula with Gödel number *sub*(y, 13, y).

53

* Nagel & Newman use "$(x)\ \neg$" instead of "$\neg\ \exists x$"

53

$\neg\ \exists x$ Proof(x, *sub*(y, 13, y))

- This arithmetical formula corresponds to the meta-mathematical statement that
  - … there exists no sequence of formulae that constitutes a proof of the formula with Gödel number *sub*(y, 13, y).
  - … or in other words, that formula (whose precise identity will obviously depend on the value substituted for "y") is not provable.
- Now consider the arithmetical formula above, and suppose it has the Gödel number n.

54

54

Turing on Computability and Intelligence

Gödel's Magical Move

- We have that:

  $g[\neg \exists x \text{ Proof}(x, sub(y, 13, y))] = n$

- Now consider the formula:

(*G*) $\neg \exists x \text{ Proof}(x, sub(n, 13, n))$

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:

$g[\neg \exists x \text{ Proof}(x, sub(n, 13, n))] = sub(n, 13, n)$

55

55

(*G*) $\neg \exists x \text{ Proof}(x, sub(n, 13, n))$

- This arithmetical formula corresponds to the meta-mathematical statement that
  - …there exists no sequence of formulae that constitutes a proof of the formula with Gödel number *sub*(n, 13, n).
  - … or in other words, that formula is not provable.
- But this arithmetical formula *itself* has the Gödel number *sub*(n, 13, n)!
- So *G* corresponds to the meta-mathematical statement that *G* itself is unprovable!!

56

56

The Denouement

- The arithmetical proposition *G* encodes the statement that *G* itself is unprovable (within the formal system chosen).
- So if *G* were <u>false</u>, then it would follow (from this encoding) that *G* was <u>not</u> unprovable:
  - Hence *G* would have to be provable in the system.
  - And if *G* were provable within a correct axiomatic system of arithmetic, then *G* would have to be <u>true</u>.
- Hence if *G* were <u>false</u>, we would have a contradiction. So *G* must be <u>true</u> and, given what it encodes, must also be unprovable within the formal system.

57

57

To spell this out further, the encoding means that:

G is true $\leftrightarrow$ *G* is unprovable

$\therefore$ $\neg$ *G* $\rightarrow$ *G* is provable

But if *G* is provable and the system provides a faithful and consistent representation of arithmetic, then *G* must be true, so we have:

$\neg$ *G* $\rightarrow$ *G* is provable $\rightarrow$ *G*

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

58

58

The *Entscheidungsproblem* Remains

- Gödel left the *Entscheidungsproblem* as unfinished business:
  - He 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 <u>*effectively computable*</u> decision procedure which would infallibly, in a finite time, reveal whether any given proposition *P* is, or is not, provable.

59

59

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 is how Alan Turing came to invent what is now known as the *Turing machine*.

60

60

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 3

"On Computable Numbers": Turing's 1936 Paper



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

61

Effective Computability

- The key contribution of Alan Turing's 1936 paper
  "On Computable Numbers, with an application to the *Entscheidungsproblem*"
  (Proceedings of the London Mathematical Society, 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.

62

62

An Outline of the 1936 Paper

- "On Computable Numbers …"
  - Turing defines the concept of a "computable number", focusing on binary fractions or what we might call "binimals" (e.g. 0.10101010 …).
  - He will show that these are more extensive than the algebraic (and hence rational) numbers, but less than the real numbers.
  - In particular, they can be proved to be *enumerable*, because the machines that generate them are enumerable.

63

63

§ 1 Computing Machines (p. 68)

- Introducing the Turing machine
  - Most of (the short) § 1 is concerned with explaining the structure of a Turing machine. But first …
- 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 outline why any finitely calculable number should be calculable by a Turing machine …

64

64

A General Model of Computation

- In § 9, Turing discusses the nature of computation in general, explaining why the Turing machine is a suitable model:

  "Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares … 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 …" (Petzold, pp. 190-1)

65

65

- Note here that the "computer", for Turing, is the *person* performing the computation:

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

66

66

Turing on Computability and Intelligence

- The computer's repertoire of operations is precisely circumscribed in terms of "simple operations" that cannot be further divided:

  "Let us imagine the operations performed by the computer to be split up into 'simple operations' … Every such operation consists of some change of the physical system consisting of the computer and his tape. … We may suppose that in a simple operation not more than one symbol is altered. … We may … without loss of generality, assume that the squares whose symbols are changed are always 'observed' squares." (Petzold, p. 192)

67

67

"Besides these changes of symbols, the simple operations must include changes of distribution of observed squares. The new observed squares must be immediately recognisable by the computer. … it is reasonable to suppose that they can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount … say … *L* squares …

It may be that some of these changes [i.e. of symbol or of observed squares] necessarily involve a change of state of mind."
(Petzold, p. 192-3)

68

68

"The most general single operation must therefore be taken to be one of the following:

(A) A possible change of symbol together with a possible change of state of mind.

(B) A possible change of observed squares, together with a possible change of state of mind.

The operation actually performed is determined, as has been suggested [*highlighted passage in slide 66*], 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." (Petzold, p. 193)

69

69

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

70

70

- Having sketched this general model of a computing machine, based on an abstract understanding of what a human computer can do, Turing points out – without proof (but correctly) – that any number computable by such a machine can also be computed by a machine of the type defined earlier (in § 1 and § 2).
- Those machines are a subset of the § 9 machines in which *B* (the maximum number of squares scanned at any one time) is one, and *L* (the maximum distance between squares scanned in successive operations) is also one.

71

71

The Turing Machine comprises:

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

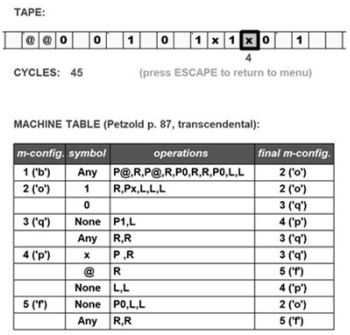
72

72

Turing on Computability and Intelligence

A Turing Machine in Action

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



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')

73

73

The "Configuration" (pp. 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")
  - the "scanned symbol"

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

74

74

The Repertoire of Actions (pp. 70-1)

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

75

75

§ 2 Definitions (p. 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")

76

76

E/*F* Squares, and Marking (pp. 93-4)

- In § 3 Turing explains that his machines will write figures only on alternate squares:
  - These are called "*F*-squares" (*figure*);
  - The others are "*E*-squares" (*erasable*), and will be used for working notes.
  - "If a symbol 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).

77

77

The Computed Number (p. 73)

- If an automatic machine starts *from a blank tape in its initial state*, then the sequence of *figures* (i.e. "0" and "1") that it writes on the tape is called the "*sequence computed by the machine*".
- "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.

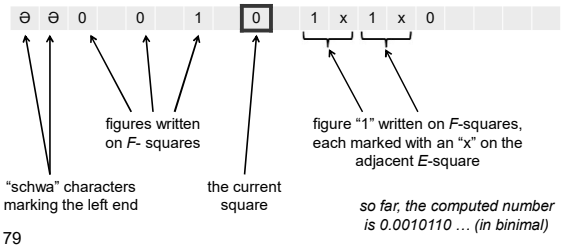
78

78

Turing on Computability and Intelligence

An Example of a Marked Tape (p. 89)

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



79

79

Circular and Circle-Free Machines (p. 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 (so sequences that correspond to rational numbers recur for ever).
  - Later, Turing will call a number that defines such a circle-free machine a "*satisfactory*" number (§ 8).
- A Turing machine that stops writing out binary digits at any stage is called "circular". [Think of it as getting stuck in a non-writing loop or "circle".]

80

80

The Complete Configuration (p. 75)

- We have already seen "*m*-configuration" (i.e. *state*) and "configuration". But Turing defines yet another use of the word, intended to capture the complete state of a computation at any time:

  "At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the m-configuration will be said to describe the *complete configuration* at that stage. The changes of the machine and tape between successive complete configurations will be called the *moves* of the machine."

81

81

§ 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:

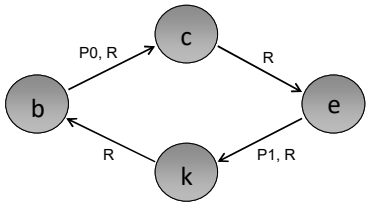
  0 1 0 1 0 1 0 1 0 1 …
- In the full form (Petzold p. 81), this requires 4 states (starting from "b" for "begin"):
  - b ("b") Prints a "0" and moves right into state c
  - c ("c") Moves right into state e
  - e ("e") Prints a "1" and moves right into state k
  - k ("k") Moves right into state b

82

82

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



83

83

Simplifying the Tables

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

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

84

84

Turing on Computability and Intelligence

A More Complex Example

- 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 …

  (see Petzold pp. 85-90 for details)
  - 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.

85

85

Listing Complete Configurations

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

    b : Ə Ə o 0 0 : Ə Ə q 0 0 : …
  - This format is flagged as "(C)", cf. Petzold p. 144.

86

86

Getting Used to Turing Machines

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

87

87

§ 4 Abbreviated Tables (pp. 113-5)

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

88

88

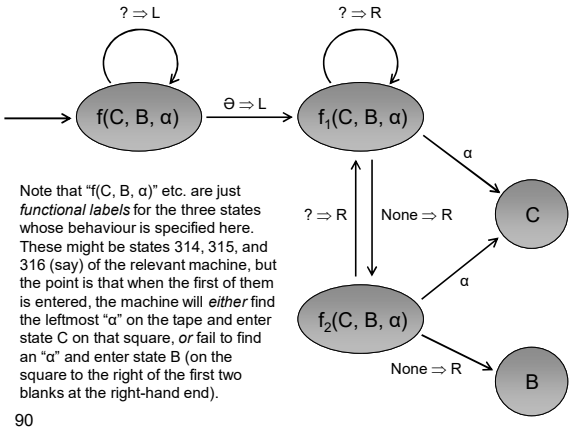
Turing's First *m*-function "Find" (p. 115)

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

89

89



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).

90

90

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

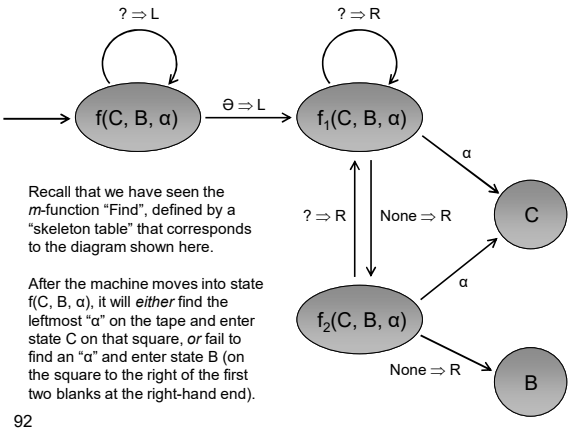
Lecture 4

Enumerating the Computable Numbers, and the Universal Turing Machine

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



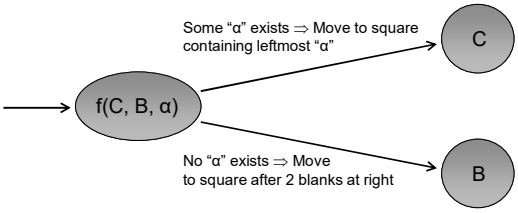
91



92

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:



93

93

"Erase" (3 arguments, erase once)

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

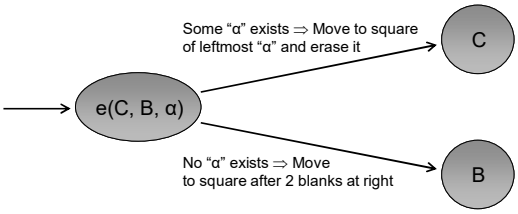
Turing (p. 118) defines two versions of this *m*-function, first with three arguments (shown above), then with two. On going into state e(C, B, α), the machine transitions into state f(e_1(C, B, α), B, α), which searches for the leftmost "α" and enters state e_1(C, B, α) on the square containing that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

94

94

Summarising "Erase"

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



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

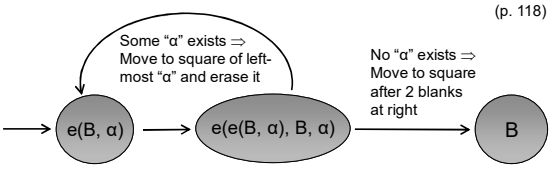
95

95

"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.

96

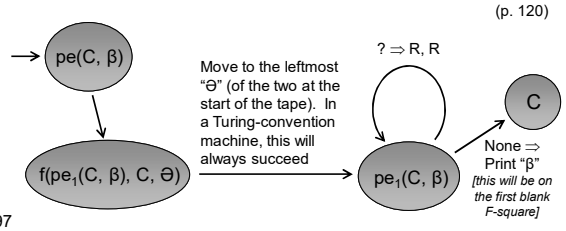
96

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)



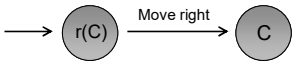
97

97

"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".

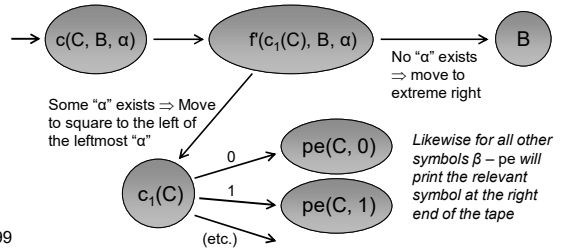
98

98

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)



99

99

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"!*

100

100

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"

101

101

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

102

102

Turing on Computability and Intelligence

§ 5 Enumeration of Computable Sequences

- Turing now explains (pp. 131 ff.) how to put all possible machine tables into a standard form, ultimately reducing each to a single number.
  - We convert each table into a table that has one write and one move/nomove (L/R/N) per line;
  - We number all the states (q_1, q_2 etc.), and all the symbols (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.

103

103

- 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 (p. 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."

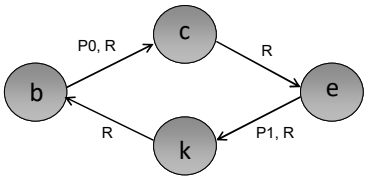
104

104

Finding the Description Number of a Machine: Turing's Example (pp. 138-40)

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

This is the machine (p. 81, slide 83) that prints out the binimal "0 1 0 1 0 1 …":



105

105

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 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

106

106

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).

107

107

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

108

108

Turing on Computability and Intelligence

Having explained his two illustrative tables, Turing continues (p. 139):

"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.]

109

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
313325311731133531173111332253111173111133531731323253117"

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

110

109

110

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

111

§ 6 The Universal Machine

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

  If this machine $\mathscr{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathscr{M}$, 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 $\mathscr{M}$."

112

111

112

Recalling Machine II

- Turing now refers back to Machine II of § 3 (pp. 87-92, slides 85-6), 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. p. 84 and slide 84), whereas at this point in the paper Turing is restricting his attention to simpler "standard form" machines (p. 132, slide 103).
  - Ignore this detail, because the process Turing is about to illustrate using Machine II and format "(C)" will work with standard form machines.

113

Generating Complete Configurations

- Recall from slide 86 how Turing lists sequences of "complete configurations" that include the entire non-blank portion of the tape, with the state and position indicated, e.g. (p. 92):

  b : Ə Ə o 0 0 : Ə Ə q 0 0 : … (C)
- At pp. 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*.

114

113

114

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 (p. 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.

115

115

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.

116

116

Identifying the Configuration

- Here the configurations in (C_1), as generated by [non-standard] Machine II (p. 87), are underlined:

  "*DA* : *DCCCDCCCDAADCDDC* :
  DCCCDCCCDAAADCDDC : …"

  (this sequence is built up at the right of the tape)

  – Recall that the complete configurations are separated by colons, and within them, just one state (represented by "D" followed by a sequence of "A"s) will appear, followed by the "scanned symbol" on the current square (represented by "D" followed by a sequence of "C"s).

  – You may find it helpful to refer to Turing's text on p. 151.

117

117

Identifying the Matching Rule

- We can underline the "trigger" configurations within the standard description of Machine I (pp. 81, 139-40) *(note we can't do this for Machine II, as it's non-standard):*

  "*DADDCRDAA;DAADDRDAAA;*
  DAAADDCCRDAAAA;DAAAADDRDA"

  (this is put at the left of the tape to start with)

  – Overall, the first quintuple has been translated thus:

Initial state	Read symbol	Write symbol	Move	Final state
b	None	0	R	c
q_1	S_0	S_1	*R*	q_2
DA	*D*	*DC*	*R*	*DAA*

118

118

Achieving Universality

- To mimic $\mathscr{M}$'s generation of a computable number, we also have to print out at each stage any *new* figures (i.e. "0"s and "1"s) produced by the transition. These will be colon-separated, between the successive complete configurations (pp. 147-8).
- If $\mathscr{M}'$ has been designed appropriately, then:

  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.

119

119

§ 7 Detailed Description of the Universal Machine

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

120

120

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 5

Settling Hilbert's Entscheidungsproblem, and the Halting Problem

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



121

§ 8 Application of the Diagonal Process

- Recall from § 5 that Turing has shown that the computable numbers are enumerable.
- But he now remarks that Cantor's diagonal argument might seem to prove otherwise.
- If the computable numbers are enumerable, then we should be able to form a list that contains them all (they are all, of course, endless binary fractions of "0"s and "1"s).

122

122

- 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 (see p. 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.

123

123

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?

124

124

- The answer is that to complete the diagonal process and compute β, we have to do more than identify 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" (pp. 177-8).

125

125

Where the Attempt Fails

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

126

126

Turing on Computability and Intelligence

Another "Diagonal" Argument

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

127

127

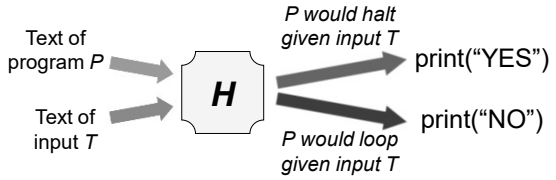
The Halting Problem

- 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?*
- *Turing's argument implies that H is impossible (though of course his own machines never halt if "satisfactory" – see Petzold pp. 328-9).*

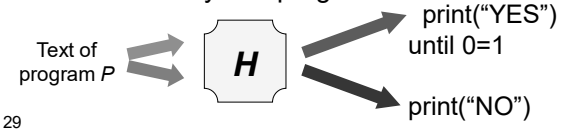
128

128

- Suppose we have such a procedure *H:*



- We now modify that program:



129

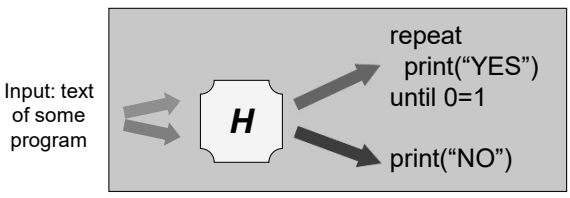
129

- 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.

130

130

The Contradictory Program *Q*



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

131

131

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)." (p. 183)
- The proof follows the familiar pattern, showing that if $\mathscr{E}$ *were* possible, then a diagonally impossible machine (namely, a circle-free testing machine) would also be possible.

132

132

Turing on Computability and Intelligence

- Suppose we have our 0-testing machine $\mathscr{E}$, to be applied to some arbitrary machine $\mathscr{M}$.
- We then consider machines $\mathscr{M}_1$, $\mathscr{M}_2$, … $\mathscr{M}_n$, …, 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" – this can be done *mechanically* by adding more states).
- Now we create another machine $\mathscr{G}$ from $\mathscr{E}$, which operates by mechanically testing $\mathscr{M}$, $\mathscr{M}_1$, $\mathscr{M}_2$, … in turn and 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:

133

133

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.

134

134

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" (p. 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 $\mathscr{E}$."

135

135

§ 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:

<u>The Church-Turing Thesis</u>

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

136

136

Turing's First Argument for the Church-Turing Thesis

- Turing (§ 9, p. 190) proposes three types of arguments for his claim about the generality of Turing machines (the third involves outlining "large classes of numbers which are computable", and constitutes § 10 – see slide 145 below).
- The first type of argument (pages190-4) maintains that a Turing machine can compute any number that would be computable by a human following a definite process. This was anticipated back in § 1, at p. 68 (see Lecture 3, slides 65-71).

137

137

Turing's Second Argument

- Turing's second type of argument (if not the particular variant he gives in this paper) has almost certainly been the most influential.
  - Here he first argues (pp. 221-8) that "If the notation of the Hilbert functional calculus [i.e. first-order predicate logic] is modified so as to be systematic, and so as to involve only a finite number of symbols, it becomes possible to construct an automatic machine, $\mathscr{K}$, which will find all the provable formulae of the calculus."

138

138

Turing on Computability and Intelligence

A "British Museum Algorithm"

- Turing sketches how a Turing machine can be defined to generate in turn all provable formulae of a system of axioms expressed in predicate formulae.
  - As Petzold explains (pp. 220-1), this is known as a "British Museum" algorithm.
- Turing then briefly points out (p. 229) that any Turing machine can be defined in terms of predicate formulae. (This is demonstrated in far more detail in § 11.)

139

139

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

140

140

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.

141

141

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" (p. 290)
  - See Petzold Chapter 15 for much more on the Lambda Calculus etc.; also pp. 325-31.
- In Volume 2 of *The Journal of Symbolic Logic* (1937), Turing's "Computability and λ-definability" provided a more rigorous proof. The new paper starts as follows:

142

142

"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 and developed by Kleene. It is shown [below] that every λ-definable function is computable and that every computable function is general recursive." (Petzold, p. 298)

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

143

143

Assessing the Church-Turing Thesis

- It seems hard in principle – and maybe even philosophically dubious – to attempt to <u>*prove*</u> 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 it remains controversial.

144

144

Turing on Computability and Intelligence

§ 10 Examples of Large Classes of Numbers which are Computable

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

145

145

§ 11 Application to the *Entscheidungsproblem*

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

146

146

- Turing refers to Hilbert's presentation of "the restricted functional calculus" (i.e. what we call predicate logic without identity). Turing then presents his goal as follows:

  > "I propose, therefore, to show that there can be no general process for determining whether a given formula $\mathscr{U}$ of the functional calculus **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." (p. 260)

- He explains (pp. 261-2) that this result is quite different from Gödel's, as we saw in the second lecture (slide 59).

147

147

From Machines to Predicate Logic

- As Turing remarks (p. 262), the proof appears "somewhat lengthy [but] The underlying ideas are quite straightforward."
- As illustrated in the next slide, he defines predicates adequate to describe the configuration and behaviour of Turing machines (pp. 263-7).
- Using these predicates (or "propositional functions"), he defines, given any machine $\mathscr{M}$, his key formula Un($\mathscr{M}$), expressing the implication from that machine's specification to the consequence that a zero will be printed (p. 267). By the earlier lemma (slides 132-5), this will in general be undecidable.

148

148

Three Predicates: Scan/State/Symbol

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

149

149

Expressing Turing Machine Rules

- These predicates enable us to specify any complete configuration of a Turing machine, in terms of the step number (or "cycle") of that configuration.
- The Turing Machine rules (i.e. quintuples or machine table) specify what should happen in moving from one cycle to the next. The next few slides explain how Turing represents this (and can usefully be read together with p. 265-6). To do so, he also needs a predicate for numerical succession (which he had introduced earlier at p. 225, in preparation for discussing Peano's axioms):
- $F(x, y)$ p. 264
  y is the immediate successor of x (i.e. $y = x + 1$).

150

150

Turing on Computability and Intelligence

- 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 $\quad I(x, y) \;\&\; K_{q_i}(x) \;\&\; R_{S_j}(x, y)$

Cycle x' $\quad I(x', y') \;\&\; K_{q_l}(x') \;\&\; R_{S_k}(x', y)$

Scanned square ↑ State ↑ Symbol ↑

151

151

- 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 (which Turing abbreviates as "Inst $\{q_i\, S_j\, S_k\, L\, q_l\}$", p. 265) to express that quintuple:

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

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

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

$\;\rightarrow$ $\quad$ THEN

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

$\&\; (z)\, [\, F(y', z)$ $\quad$ & $\forall z$ [<u>either</u> z=y (i.e. y'+1)

$\;\vee\, ([R_{S_0}(x, z) \rightarrow R_{S_0}(x', z)] \;\&$ $\quad$ <u>or</u> square z keeps

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

152

152

- 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 (p. 266):

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

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

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

$\;\rightarrow$ $\quad$ THEN

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

$\&\; (z)\, [\, F(z, y')$ $\quad$ & $\forall z$ [<u>either</u> z=y (i.e. y'-1)

$\;\vee\, ([R_{S_0}(x, z) \rightarrow R_{S_0}(x', z)] \;\&$ $\quad$ <u>or</u> square z keeps

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

153

153

- 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') $\quad \forall\, x, y, x'$, and y'

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

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

$\;\rightarrow$ $\quad$ THEN

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

$\&\; (z)\, [\, F(z, y')$ $\quad$ & $\forall z$ [<u>either</u> z=y (i.e. y'-1)

$\;\vee\, ([R_{S_0}(x, z) \rightarrow R_{S_0}(x', z)] \;\&$ $\quad$ <u>or</u> square z keeps

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

154

154

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 (pp. 267, 311), 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 148 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.

155

155

Peano's Axioms

- A complication here is that Turing needs to specify the relevant properties of the number sequence (corresponding to the sequence of configurations).
- These properties are captured by *Peano's Axioms*, of which this is one version (see Petzold p. 223):
  1. Zero is a number.
  2. Every number has a successor that is also a number.
  3. Zero is not the successor to any number.
  4. Two numbers that are the successors to the same number are equal.
  5. If $P(0)$ is true, and if $P(x)$ implies $P(\text{successor}(x))$ for all numbers x, then $P(x)$ is true for all numbers x.

156

156

Turing on Computability and Intelligence

- Turing's original paper slips up in defining the Peano conditions (pp. 226-7, 267-8). He provided a correction (pp. 268, 311), which uses Q as an abbreviation for:

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

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.

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

157

157

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

- The formula Un($\mathscr{M}$) for machine $\mathscr{M}$, which is in general undecidable, is as follows (pp. 268, 311):

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

  - 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.

158

158

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 (pp. 269-76): "If S_1 appears on the tape in some complete configuration of $\mathscr{M}$, then Un($\mathscr{M}$) is provable".
  - Lemma 2 (pp. 276-77): "If Un($\mathscr{M}$) is provable, then S_1 appears on the tape in some complete configuration of $\mathscr{M}$".

159

159

- 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 (p. 271).
- Here he uses an earlier abbreviation (from p. 268), with "A($\mathscr{M}$)" standing for:

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

- 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" (p. 272: base case pp. 272-3; induction step pp. 273-5).

160

160

- 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 (p. 277) from substitution of the relevant propositional functions in Un($\mathscr{M}$).
- Thus (p. 277), referring back to the lemma of § 8, "We are now in a position to show that the *Entscheidungsproblem* cannot be solved".

161

161

- To sum up, the lemma of § 8 proved that no machine can be constructed which will reliably determine whether any specified machine will ever print a "0".
- But since that behaviour (printing a "0", or not doing so) can be captured by a predicate logic formula Un($\mathscr{M}$), it follows that no machine can be constructed which will reliably determine whether or not that formula is provable.
- Hence Hilbert's *Entscheidungsproblem* – to devise a mechanical procedure to determine whether or not *any* predicate logic formula is provable, has no solution. QED!!

162

162

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 6

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



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

163

Useful Books

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

164

164

Useful Collections of Papers

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

165

165

Useful Web Resources

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

166

166

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

- Robin Gandy, Turing's 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)

167

167

§ 1 The Imitation Game

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

168

168

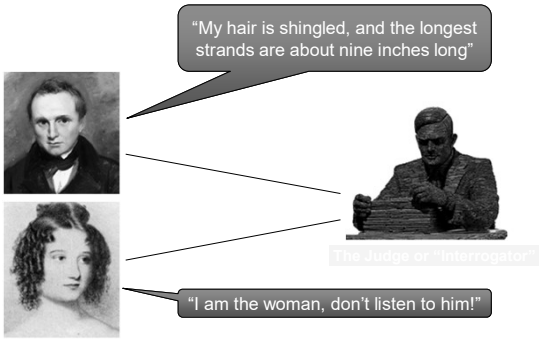
Turing on Computability and Intelligence

The Gender Imitation Game

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

169

169



- The man pretends to be a woman. If the interrogator can't reliably distinguish them from their verbal responses, then the man succeeds.

170

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 …

171

171

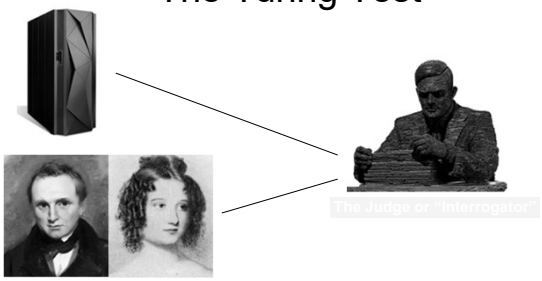
The Computer Imitation Game

- "We now ask …, 'What will happen when a machine takes the place of *A* (the deceitful man) in this game?' Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, 'Can machines think?'"
- The relationship between these questions and the original is left obscure. And it's not clear (until § 2 and § 5) that the computer's role is to imitate a *human* ("man") rather than specifically a woman.

172

172

The Turing Test



- If the interrogator can't reliably distinguish the computer from the human, then the computer program must be judged to be *intelligent*.

173

§ 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".

174

174

Turing on Computability and Intelligence

Flexibility of the Q&A Format

Q: Please write me a sonnet on the subject of the Forth Bridge.

A: *Count me out on this one. I never could write poetry.*

Q: Add 34957 to 70764.

A: (Pause about 30 seconds and then give as answer) *105621.*

Q: Do you play chess?

A: *Yes.*

Q: I have K at my K1, and no other pieces. You have only K at K6 and R at R1. It is your move. What do you play?

A: (After a pause of 15 seconds) *R-R8 mate.*

175

175

Bias in Favour of Human Thought?

- But Turing notes the possible objection: "May not machines carry out something which ought to be described as thinking but which is very different from what a [human person] does?"
- He responds: "This objection is a very strong one, but at least we can say that, if, nevertheless, a machine can be constructed to play the imitation game satisfactorily, we need not be troubled by this objection."
  - This seems to be suggesting that *success* in the imitation game is *sufficient* to prove intelligence, but not a *necessary* condition.

176

176

The *Viva Voce* Example

- Later, in § 6 (4), Turing will give an example of a *viva voce* examination, concerning the choice of words in a poem, which seems quite convincing as a demonstration of intelligence.
  - *If* conversation of this level of sophistication and focus were to be produced by a computer over a wide range of topics, in a way that involved appropriate meaning-sensitive processing and without any form of "cheating" (e.g. use of "canned" responses"), it might indeed be hard to deny that the machine was demonstrating "intelligence".

177

177

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.

178

178

§ 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", 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.

179

179

§ 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." – here we see clear echoes of the 1936 paper.
- Turing then gives an outline of how a digital computer might operate, remarking that "they can in fact mimic the actions of a human computer very closely".

180

180

Turing on Computability and Intelligence

Varieties of Digital Computer

- Turing considers some particular cases:
  - A "digital computer with a random element", 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, thus illustrating that they can be mechanical and need not be electrical.

181

181

§ 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"), 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."
- Thus digital computers "are *universal* machines."

182

182

The Programming Question

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

183

183

§ 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."
- 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".

184

184

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

185

185

(1) The Theological Objection

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

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

186

186

Turing on Computability and Intelligence

(2) The "Heads in the Sand" Objection

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

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

187

187

(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".
  - They may fail on questions of the type:
    "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.

188

188

(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 ..."

189

189

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."

190

190

- Turing goes on to give his *viva voce* example about the sonnet (picking up on Jefferson's reference to sonnet composition):
  - "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."

191

191

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.

192

192

Turing on Computability and Intelligence

Two Distinct Lines of Thought

- There are two quite different lines of thought here, which Turing would have done well to distinguish:
  - Jefferson is "denying the validity" of the Turing test *because it does not test for genuine consciousness*, and genuine consciousness (rather than "artificial signalling") is necessary for intelligence.
  - "Artificial signalling" of apparent emotions is unworthy of being deemed intelligent *because it is an "easy contrivance"*.

193

193

A Better Response

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

194

194

(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*.
- The limited machines of 1950 could not do these, but it requires argument to show than *none* could.
- Several of these alleged limits seem just to take us back to the argument from consciousness.

195

195

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?' ..."
- 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!

196

196

(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".
- In response, Turing points out that:
  - We are fallible, and can often 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.

197

197

- 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

198

198

Turing on Computability and Intelligence

(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."

- 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.

199

199

(8) The Argument from Informality of Behaviour

- Turing reduces this argument to:

  "If each man had a definite set of rules of conduct by which he regulated his life he would be no better than a machine. But there are no such rules, so men cannot be machines."

  – He points out that formally it is a fallacy ($R \rightarrow M$ is not the same as $M \rightarrow R$).

  – Moreover it is hard to establish that we are *not* in fact governed by "laws of behaviour" (to be distinguished from "rules of conduct")

200

200

(9) The Argument from Extra-Sensory Perception

- Turing's sympathy towards telepathy, clairvoyance, precognition and psycho-kinesis might seem surprising: "the statistical evidence, at least for telepathy, is overwhelming".
  – Few would now agree with this, but Turing was apparently impressed with the work of J. B. Rhine (see Hodges' biography, p. 416).
  – Turing seems also to assume that if extra-sensory perception were possible, no machine could mimic this, so we might need a "telepathy-proof room".

201

201

§ 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."
- 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."

202

202

- After a discussion of technological needs (storage of around 10^9, speed around 1,000 times faster than nerve cells), he focuses on the problem of programming a machine to play the game successfully.
- The solution, he suggests, may be to try to simulate a baby's mind rather than an adult's, and provide it with the ability to learn.
  – Here Turing's discussion seems quite unrealistic, but he did not have the advantage of our further 60 years of experience, which have shown how difficult it is to implement general learning systems to compete with the products of evolution.

203

203

- One important point Turing makes here is that a learning machine is highly likely to behave in ways that its programmers could neither foresee nor understand, and also to make "mistakes". Trial and error learning, in particular, requires some element of randomness.
- "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."
- On this uncertain note, the paper ends.

204

204

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 7

Blockhead, the Chinese Room, and ELIZA



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

205

BlockHead

- In 1981, Ned Block attacked the Turing Test on the grounds that it can be passed by a *mindless* machine.
  - His postulated machine, widely known as "Blockhead", works by storing every possible "sensible" conversation of a given length, and choosing its responses accordingly.
  - Something working so mindlessly, by a mechanism that involves no understanding, surely cannot be called *intelligent*, but it will always generate sensible conversation!

206

206

A Reality Check

- How many "sensible" conversations are there, involving 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 $\approx 100^{200} = 10^{400}$.
  - 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)!

207

207

An Important Distinction

- Imagine a system that exhibits some impressive behaviour. In attributing *intelligence* to it, we could mean either:
  - That behaviour is *definitive* of intelligence: anything that behaves like that is *ipso facto* correctly described as "intelligent".
  - That behaviour provides strong *evidence* of intelligence, because it could only plausibly be generated by something with the capacity for sophisticated information processing (etc.).

208

208

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.

209

209

- 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.

210

210

Turing on Computability and Intelligence

Thought-Experiments as "Intuition Pumps"

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

Daniel Dennett, in *The Third Culture*, ed. Brockman

211

211

- 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 easily suggest quite different conclusions …

212

212

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 – provide meaningful and appropriate answers to those questions (such as might be produced by a competent and intelligent native speaker of Chinese).

213

213

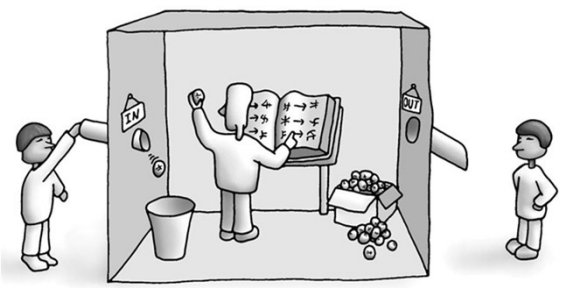
- But the man inside the room has no knowledge whatever of the Chinese language or of the *semantics* – the meaning or significance – of the symbols he is reading or writing.
- Instead, he is generating his written "answers" by strictly applying rules based purely on the *syntax* – the shape and structure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

214

214

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 …



215



216

Turing on Computability and Intelligence

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!).

217

217

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)

218

218

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.)

219

219

The Cautious Interpretation

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

220

220

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)

221

221

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.

222

222

Turing on Computability and Intelligence

The Robot Reply

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

223

223

"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)

224

224

Searle *versus* Turing

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

225

225

Contrary Intuitions

- Turing: "Imagine a computer program that is able to converse like this …
  ... how could you possibly deny that it is genuinely intelligent?"
- Searle: "Imagine a computer program that conducts its conversation using crudely syntactic processes like this …
  ... how could you possibly claim that it is genuinely intelligent?"

226

226

Back to Reality

- As in the case of Blockhead, we can inject some realism by asking how plausible the two thought-experiments are in practice.
  - The Chinese Room (especially in its later unrestricted version) is as wildly implausible as Blockhead, supposing sophisticated linguistic behaviour generated in real time by manually consulting books of rules contained within a room (with no scope for sensory input, real-time updating, or emotional reaction etc.).
  - Turing's predictions seem more realistic …

227

227

Turing's Predictions from § 6

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

228

228

Turing on Computability and Intelligence

A Correct Prediction?

- So by 2000 "one will be able to speak of machines thinking without expecting to be contradicted". And indeed by 2000 this conversation was plausible:

  "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."

229

229

Another Plausible Prediction, but …

- Turing's other prediction is that by 2000:

  "… it will be possible to programme computers … to make them play the imitation game so well that an average interrogator will not have more than 70 per cent. chance of making the right identification after five minutes of questioning."

  This again looks plausible in retrospect. *If sustained and systematic research effort had been directed towards this goal*, it seems quite likely that it could have been achieved by 2000.

  — However the prediction is plausible for a bad reason!

230

230

… an Unreliable Criterion

- Turing's words naturally suggest a criterion for progress towards machine intelligence, based on how well the program can fool an "average interrogator", and for how long.
- This has set the pattern for subsequent attempts at "passing the Turing Test" (as illustrated in the next slide), but this is highly regrettable, because

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

231

231

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.

232

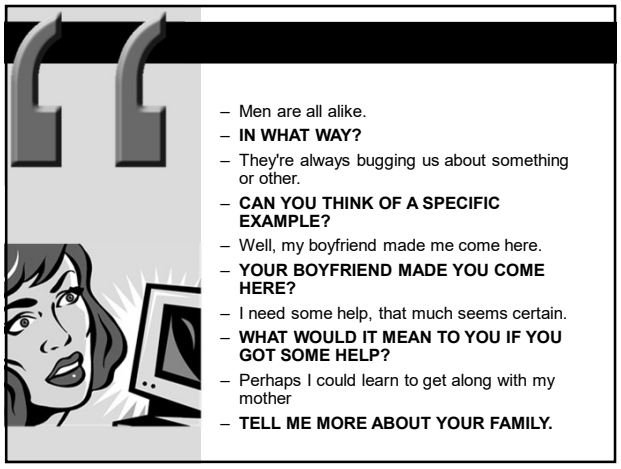
232

Weizenbaum's *ELIZA*

- In 1966, Joseph Weizenbaum published his *ELIZA* program and "DOCTOR" script, showing how very simple text manipulation could generate a plausible conversation.
  - The "DOCTOR" plays the role of a Rogerian psycho-therapist, largely echoing what the human says, expressing sympathy etc., and asking gentle questions to elicit their feelings.
  - Computer responses are generated by making small changes to the human inputs, exchanging first- and second-person etc.

233

233



234

Turing on Computability and Intelligence



- My mother takes care of me.
- **WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?**
- My father.

…

- **WHAT ELSE COMES TO YOUR MIND WHEN YOU THINK ABOUT YOUR FATHER?**
- Bullies.
- **DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE?**

235

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?"

236

236

Elizabeth: *ELIZA*'s Cleverer Sister

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

237

237

Chatbots are NOT Intelligent!

- Experimentation with chatbots (e.g. the ELIZA script available within *Elizabeth*) can quickly strengthen one's impression that Searle is right: there is no *semantics* here, only blind manipulation of *syntax*.
- The script that runs "Eugene Goostman", the chatbot that supposedly "passed the Turing Test" in 2014, follows the standard ELIZA-style pattern. Faced with focused questioning, it soon reveals its stupidity …

238

238



```
<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>
```

239

239

Where do you live?

I am from Ukraine, from the city called Odessa. You might have heard about it.

I have. Were you there on May 2?

Was I? I would rather not talk about it if you don't mind. Wonna ask me something more?

I want to talk about this. Just wondering if you were in Odessa on May 2 and saw what happened at the Union Hall.

I think you can answer your question yourself because it is rhetorical. Could you tell me about your job, by the way?

I am a journalist.

Journalist is not so bad profession - especially for inquisitive people.

I am inquisitive. So did you see the fire at Union Hall on May 2?

What a stupid questions you ask! Did I forget it or you really haven't told me about the place you are from?

…

So, about that fire in Odessa on May 2: Did you see it?

Odessa is a very big city - a kind of "Babylon," because there are lots of nationalities living there - Ukrainians, Russians, Jews, Greeks and many many more.

240

Leonid Bershidsky, on BloombergView

240

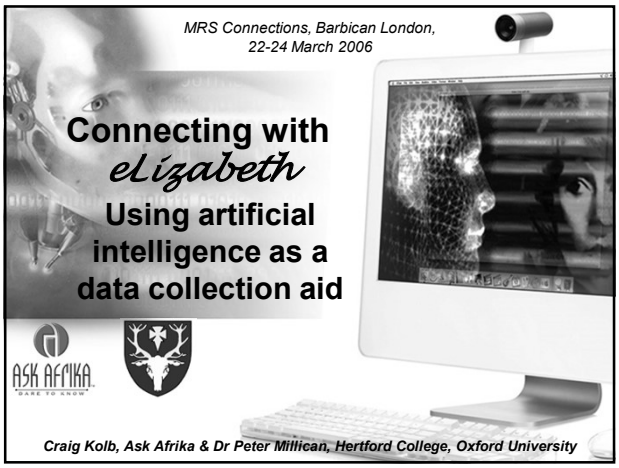
Turing on Computability and Intelligence

Uses for Chatbots

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

241

241



242

The Problem with the Turing Test

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

243

243

The Turing Test – Verdict So Far

- Implausible when interpreted as fooling "the average interrogator" for a modest time.
  - This is achievable by ELIZA-style pretence and misdirection, rather than sophisticated and explicit processing of information.
- More plausible as a ***sufficient*** condition for intelligence when interpreted more stringently (cf. Turing's sonnet conversation).
  - But then seems *inappropriately* demanding in terms of indistinguishability from a human.

244

244

Rejecting Anthropocentrism

- As we saw before (slide 176), Turing himself acknowledges that a machine might "think" in ways very different from a human.
- Moreover mimicking "subcognitive" human responses is likely to be extremely difficult.
  - Robert French (1990) suggests questions like: "Rate 'Flugly' as the name of a glamorous model or a cuddly toy."
  - But testing for indistinguishability of this sort – in tastes and instinctive reactions – seems *inappropriate* when testing for *intelligence*.

245

245

The *Tutoring* Test

- Perhaps a better form of test is one focused on *revelation* of information processing – in a *teaching* task – rather than *deception*.
  - Enables focus on a specific domain, e.g. some aspect of chemistry, which is suitably deep with complex informational structures.
  - No need to pretend, and no expectation of indistinguishability: the test is how well the system can teach the theory (e.g as well as a human?).
  - Could provide *seriously useful* products, and also stimulate progress towards "intelligent" processing.

246

246

Turing on Computability and Intelligence

Alan Turing on Computability and Intelligence

Lecture 8

Searle versus Turing: Conclusion



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

247

Summary on Turing So Far

- Turing's 50-year ***predictions*** in his 1950 paper look plausible (slides 228-30).
- However as a ***criterion of intelligence***, his proposed Turing Test is very dubious:
  - If interpreted generously (e.g. ≤70% success by an average interrogator after 5 minutes), it is undermined by the discovery that humans are easy to fool by chatbots (slides 231-40).
  - If interpreted more strictly, it is too demanding: an intelligent system need not be designed to mimic human cultural reactions etc. (slides 243-5).

248

248

Summary on Searle So Far

- Searle's Chinese Room thought-experiment is wildly implausible in terms of practicality (slides 216, 227).
  - As with "Blockhead", this casts some doubt on our "intuitive" responses to it (cf. slides 207-10).
- 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 (slide 222). We have yet to consider his answer to the Robot Reply.

249

249

An "Intelligent" Tutoring System?

- Imagine a computerised tutoring system which is capable of highly sophisticated information processing in a complex domain (slide 246).
  - It operates not by pretence, but rather by genuine responsiveness to information structures, in ways that reflect how human experts would think.
- Nevertheless, Searle will insist that the system's processing is *merely* "syntactic", with no genuine "semantics".
  - It has no real *understanding* of the domain – its "thought" cannot "reach out" to the real world.

250

250

Intrinsic/As-If/Derived Intentionality

- Searle's *The Rediscovery of the Mind* (1992, pp. 78-9) emphasises this distinction:
  - *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.

251

251

Two Responses to Searle

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

252

252

Turing on Computability and Intelligence

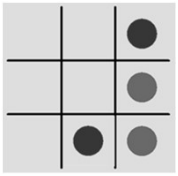
Arithmetic and Abstract Games

- Searle's objection seems quite strong in respect of information processing about physical things (e.g. chemicals or trees): "surely", we are inclined to say, "mere internal processing of formal symbols cannot *genuinely* constitute thought about *real* trees!"
- But this isn't so clear with thought about abstract entities such as numbers or chess positions, where apparently "intelligent" processing need not be responsive to the properties of concrete things like trees, but only to appropriate logical relations, which can apparently be represented formally.

253

253

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*).
- 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"?

254

254

Avoiding Philosophy of Mathematics

- Let us here put purely abstract thinking, e.g. about arithmetic and chess, to one side:
  - It is extremely obscure how even *we* manage to make reference to, and think, about abstract entities (which neither appear to our senses, nor seem to have any direct causal impact on us).
  - Searle's main point puts us *with* the animals and *against* digital computers, and thought about *physical* things is far more central to his case.
  - So rather than philosophy of mathematics, let us turn to something paradigmatically physical …

255

255

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

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

256

256

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 …

257

257

Combining Robot and System Replies

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

258

258

Turing on Computability and Intelligence

What Do Machines Lack?

- Searle will no doubt deny that the robotic lumberjack's internal states have "semantic content", despite having these real-world causal relations. But what then do they lack?
- Searle's Chinese-style arguments suggest he thinks "understanding" in the sense of *conscious awareness* is the crucial factor.

  (But note in passing that "understanding" is a very slippery word, that can itself be *understood* in a variety of different ways!)

259

259

Do Animals Have What It Takes?

- Searle seems clear that some animals have what digital computers lack:

  "Visual and auditory experiences, tactile sensations, hunger, thirst, and sexual desire, and all caused by brain processes and they are realised in the structure of the brain, and they are all intentional phenomena. … it is just a plain fact about biological evolution that it has produced certain sorts of biological systems, namely human and certain animal brains, that have subjective features."

260 *Minds, Brains, and Science* (1984), pp. 24-5, cf. 40-1

260

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

261

261

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.

262

262

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.

263

263

Three Categories of "Intelligence"?

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

264

264

Turing on Computability and Intelligence

Reminder: Turing on Consciousness

- Recall Turing's response to Jefferson at § 6.4 (slides 189-94). There he seems to accept that to support machine intelligence, he must take *machine consciousness* to be as reasonable as believing that *other people* are conscious:

  "This argument [of Jefferson's] appears to be a denial of the validity of our test. According to the most extreme form of this view ... the only way to know that [either a machine or] a *man* thinks is to be that particular man. It is in fact the solipsist point of view. It may be the most logical view to hold but it makes communication of ideas difficult. ... it is usual to have the polite convention that everyone thinks."

265

265

- Turing goes on to give his *viva voce* example about the sonnet:

  "What would ... Jefferson say if the sonnet-writing machine was able to answer like this in the *viva-voce*? ... if the answers were as satisfactory and sustained as in the above passage ..."

- But here – I suggested on slide 194 – Turing should have continued:

  "... then there would be reason to call the machine 'intelligent' irrespective of whether or not it has genuine feelings. *Intelligent thinking need not require consciousness (nor even – contra Searle – potential consciousness).*"

266

266

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.

(See also my paper, "The Philosophical Significance of the Turing Machine and the Turing Test", 2013)

267

267

Turing Predicts Conceptual Change

- Recall Turing's prediction from § 6 of the 1950 paper:

  "The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."

- Turing thus anticipated alterations in "the use of words", that is, *evolution of our conceptual scheme*. And indeed his own discoveries, deriving from his 1936 paper, give excellent reason for favouring such evolution, in the direction of separating "intelligence" from consciousness.

268

268

Real Novelty Beats Imagined Similarity

- A problem with "intuition pumps" is that they can tempt us into *misleading* or *simplistic* comparisons, based on familiar "intuitive" assumptions and drowning out unexpected differences.
- *Practical confrontation with new realities* can be far more vivid, and far more surprising, forcing us to seriously revise our thinking by making it impossible to ignore things that are *unintuitive*.
- Major scientific progress has often involved such major novelties, bringing "paradigm shifts" and fundamentally new *methods of explanation*.

269

269

Floridi on Conceptual Revolutions

- Luciano Floridi sees Turing's as the fourth in a series of conceptual revolutions concerning humanity's "fundamental nature and role in the universe":

  "We are not immobile, at the centre of the universe (Copernicus); we are not unnaturally distinct and different from the rest of the animal world (Darwin); and we are far from being entirely transparent to ourselves (Freud). We are now slowly accepting the idea that we might be informational organisms among many agents (Turing) …" (2008, p. 651)

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

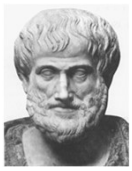
270

270

Turing on Computability and Intelligence

Modes of Explanation (until 1600)

- Purposive Design (God)
  - Things in the world (e.g. animals, plants, minerals) take the form they do because they were ***designed*** to be that way.
- Purposive Action (Aristotle)
  - Things in general (humans, stones, water, fire, planets etc.) behave as they do because they are striving to achieve some desired state, or to avoid some abhorrent situation.



271

271

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.



272

272

The Naturalistic Turn (1739-1859)

- Naturalist Psychology (Hume)
  - Human behaviour is governed more by emotions and imagination than 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.

273

273

Modern Physics (1905-30)

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

274

274

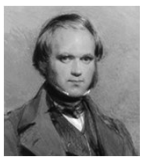
Breaking Paradigms

- 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.

275

275

Darwin, Einstein, and Hume





276

- 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."

276

Turing on Computability and Intelligence

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

277

277

Turing's Discovery (1936)



- Information processing can be understood in terms of symbolic inputs and outputs, governed by explicit and automatic processes with a limited range of operations (as defined by a "Turing machine").
- Hence information does not presuppose an "understanding" mind.
  - So since Turing, we are confronted with sophisticated *information processing* without *conscious purpose*, just as Darwin brought us sophisticated *adaptation* without *intentional design*.

278

278

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?

279

279

Open-Texture

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

280

280



"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

281

281

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!

282

282

Turing on Computability and Intelligence

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.

283

283

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).

284

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 extremely dubious)!

285

Turing's Solipsistic Mistake

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

286

286

Consciousness and its Role

- The phenomenal reality of consciousness strongly suggests that it is something causally active, not just an abstraction of information processing patterns or functional role.
  - Its evolutionary role, however, seems mysterious: "zombies" lacking consciousness but behaving exactly like us would be just as successful.
  - Nevertheless the close correlation between consciousness (e.g. of pleasures and pains) and factors affecting our wellbeing is overwhelming evidence that consciousness is indeed causally active, not just an "epiphenomenon".

287

287

Dismissing Human Solipsism

- Consciousness could only evolve in a way that is so well-tuned to our needs if it is indeed causally active, *even if we are unable to work out how*.
  - Evolutionary theory is well understood and supported by overwhelming empirical evidence. *Strong scientific evidence trumps armchair speculation!*
- We humans are all products of the same evolved biological processes. So even if we cannot work out how consciousness arises, we have very good reason to attribute it to each other.
  - But there is *no such reason* to attribute it to computers.

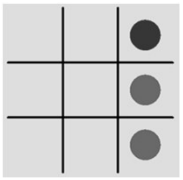
288

288

Turing on Computability and Intelligence

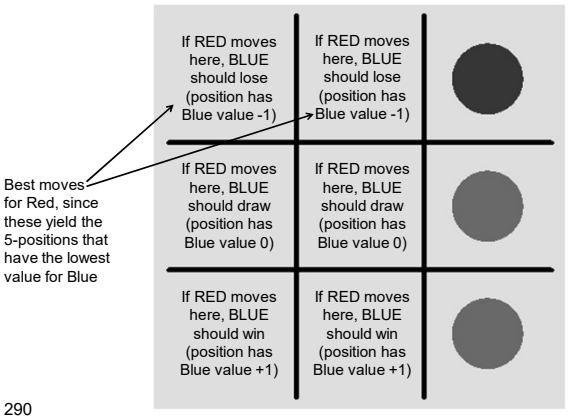
Appendix: Noughts-And-Crosses

- 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?



289

289



290

290

"Minimaxing"

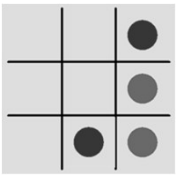
- Red winning = Blue losing (and vice-versa).
  - So the value of a position to Red is the inverse of its value to Blue (e.g. -1 as opposed to +1).
  - Hence Red tries to find moves which *maximise* the value to her, and *minimise* the value to Blue.
  - Red, faced with a 6-position, chooses the move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

291

291

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.



292

292

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.

293

293

- 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.

294

294