

On Turing, Creativity and Metabiology

Gregory Chaitin*

September 21, 2010

Abstract

Remarks on Alan Turing's oeuvre for the book *Alan Turing—His Work and Impact* edited by S. Barry Cooper and Jan van Leeuwen.

Part I: How Do We Compute? What Can We Prove?

On Computable Numbers, with an Application to the Entscheidungsproblem

In this remarkable paper Turing starts by observing that most real numbers are uncomputable—indeed this is the case with probability one. Furthermore he observes that incompleteness is a corollary of uncomputability, since if it is always possible to prove whether or not something is the case using a fixed formal axiomatic theory, then there is in principle a mechanical procedure for searching through all possible proofs and mechanically deciding the answer.

This makes incompleteness much more natural and fundamental than the assertion “I’m unprovable!” that is true if and only if it is unprovable, that was constructed by Gödel. So, following Turing 1936, we have a much bigger problem than following Gödel 1931.

On the one hand he taketh away, on the other he giveth, for although Turing shows that formal languages for mathematical reasoning are necessarily

*Chaitin is emeritus researcher at the IBM Watson Research Center, an honorary professor at the University of Buenos Aires, and a member of the Académie Internationale de Philosophie des Sciences. His email address is gjchaitin@gmail.com.

incomplete, he also shows that formal programming languages can have a kind of completeness called *universality*. No formal language can express all possible proofs, but programming languages are commonly universal, that is to say, capable of expressing essentially any algorithm.

In our *Search for the Perfect Language* (to echo Umberto Eco), let's now consider how expressive different programming languages can be. Given a particular programming language, two important things to consider are the complexity $H(x)$, namely the size in bits of the smallest program to calculate x as a function of x , and the corresponding probability $P(x)$ that a program whose bits are chosen by using independent tosses of a fair coin will compute x .

We are thus led to select a subset of the Turing universal languages that minimize H and maximize P ; one way to define such a language is to consider a universal computer U that runs self-delimiting binary computer programs $\pi_C p$ defined as follows:

$$U(\pi_C p) = C(p).$$

In other words, the result of running $\pi_C p$ on U is the same as the result of running p on C .

Any two such universal languages U and V will necessarily have

$$|H_U(x) - H_V(x)| < c$$

and

$$P_U(x) > P_V(x) \times 2^{-c}, \quad P_V(x) > P_U(x) \times 2^{-c}.$$

It is in this precise sense that such a universal U minimizes H and maximizes P .

Using such a U we can define the halting probability Ω , for example as follows:

$$\Omega = \sum P(n)$$

over all positive integers n , or alternatively

$$\Omega' = \sum 2^{-H(n)}$$

which has a slightly different numerical value but essentially the same paradoxical properties.

What are these properties? Ω is a form of concentrated mathematical creativity, or, alternatively, a particularly economical Turing oracle for the

halting problem, because knowing N bits of the dyadic expansion of Ω enables one to solve the halting problem for all programs which compute a positive integer that are up to N bits in size. It follows that the bits of the dyadic expansion of Ω are irreducible mathematical information; they cannot be compressed into a theory smaller than they are.

More precisely, it takes a formal theory of complexity $\geq N - c$ (one requiring a $\geq N - c$ bit program to enumerate all its theorems) to enable us to determine N bits of Ω . From this it follows that Ω is Borel normal, so that Ω is a particularly natural example of a normal number. In 1933 Turing's friend David Champernowne found a natural example of a number normal for blocks of all size in base-ten; Ω provably has this property in any base.¹

From a philosophical point of view, however, the most striking thing about Ω is that it provides a perfect simulation in pure mathematics, where all truths are necessary truths, of contingent, accidental truths—i.e., of truths such as historical facts or biological frozen accidents.

Indeed, I have just recently come to understand that the most important property of Ω is that it opens a door for us from mathematics to biology. The halting probability Ω contains infinite irreducible complexity and in a sense shows that pure mathematics is even more biological than biology itself, which merely contains extremely large finite complexity. For each bit of the dyadic expansion of Ω is one bit of independent, irreducible mathematical information, while the human genome is merely 3×10^9 bases = 6×10^9 bits of information.

For more on biology, see my remarks in Part IV.

Systems of Logic based on Ordinals

[See the references to Turing oracles in my remarks *On Computable Numbers* and on *Computing Machinery and Intelligence*.]

¹Andrew Hodges conjectures that Turing's work on normal numbers helped Turing to formulate the notion of a computable real; see Hodges' review of Copeland's *The Essential Turing* in the November 2006 *AMS Notices* and Turing's *A Note on Normal Numbers* in Part II. Also see the remarks on *A Note on Normal Numbers* in this book by Hodges and Verónica Becher.

Part II: Hiding and Unhiding Information: Cryptology, Complexity and Number Theory

The Word Problem in Semi-groups with Cancellation

Following Turing’s 1936 discovery of the halting problem and how to derive incompleteness from it as a corollary in *On Computable Numbers*, a great deal of work was done by many mathematicians finding “natural” versions of the halting problem in many traditional areas of mathematics, that is, in pre-Turing mathematics. Turing describes much of this work in his lovely 1954 expository paper *Solvable and Unsolvable Problems*, movingly published the year of his untimely death. Turing’s word problem paper is a complicated piece of such work that was carried out by Turing himself; obviously he wished to reassure himself that the halting problem was ubiquitous in real mathematics, which is the current consensus.²

This very large body of work, which unfortunately has never been collected in a book and remains scattered in crumbling, dusty journals, can also be used to show that the halting probability Ω is hiding in many fields of traditional mathematics. For example, using 1947 work of Emil Post it is easy to show³ that there is a semi-group without cancellation such that for each positive integer k the number of words $\lambda q_{\text{initial}} a^k b^n \lambda$ which are equal to the word $\lambda q_{\text{final}} \lambda$ is finite or infinite depending on whether the k th bit of Ω is a 0 or a 1. Following work of Toby Ord and Tien D. Kieu, 2004, one can replace “finite” or “infinite” here by “even” or “odd.”

Most unfortunately Turing did not live long enough to see the most beautiful example of the halting problem that occurs in a traditional field of mathematics, number theory, namely the 1970 Davis-Putnam-Robinson-Matiyasevich proof that asking for a general method to determine whether or not a diophantine equation has a solution is equivalent to solving the halting problem. In particular, for any computer programming language, there is a diophantine equation with a parameter k that has a solution if and only if the k th computer program in that language ever halts. Similarly, there is a diophantine equation with a parameter k that has finitely or infinitely many solutions—or alternatively an even or an odd number of solutions—

²See Stephen Wolfram’s *A New Kind of Science* and the February 2010 *AMS Notices* piece *Can’t Decide? Undecide!* by Chaim Goodman-Strauss.

³Chaitin, 2007.

depending on whether the k th bit of Ω is a 0 or a 1.

Solvable and Unsolvable Problems

This lovely paper by Turing beautifully illustrates Hilbert's remark in his 1900 Paris International Congress of Mathematicians paper on *Mathematical Problems* that one does not truly understand something until one can explain it to the first man that one meets on the street.⁴ At a more philosophical level, note that Hilbert, Turing and computer programming formalisms are often taken as the justification for extreme formalism in presenting mathematics. Emil Post, on the contrary, insisted that the work of Gödel and Turing argued against formal axiomatics and in favor of a return to meaning and truth.⁵ And here is Turing himself explaining the basic ideas behind his work without using any mathematical or programming formalism, in very clear, down to earth English with rather concrete imagery.⁶

A Note on Normal Numbers

[See my remarks *On Computable Numbers*, where I refer to normal numbers and to this paper by Turing.]

Part III: Building a Brain: Intelligent Machines, Practice and Theory.

Computing Machinery and Intelligence (the celebrated MIND article)

What is human intelligence? Is it mechanical or is it creative? In this paper Turing argues forcefully for the former, mechanical, but his *On Computable Numbers* makes a dramatic statement in favor of creativity, since he shows

⁴My paraphrase. Hilbert actually states that "An old French mathematician said: 'A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street'."

⁵See Post's 1941 remarks quoted at the end of Jeremy Gray's 2008 book *Plato's Ghost: The Modernist Transformation of Mathematics*.

⁶See also the reference to this paper in my discussion of *The Word Problem in Semi-groups with Cancellation*.

that there is no mechanical procedure even for solving something as simple as the halting problem. Let us contrast Turing's *MIND* article with his *On Computable Numbers* more forcefully. According to his piece in *MIND*, human thinking is mechanical. But *On Computable Numbers* is—as Paul Feyerabend puts it in another context—*Against Method* in mathematics, and people can do mathematics! How come?

According to Gödel, mathematicians sometimes have direct access to the Platonic world of ideas; in Turing's famous *Systems of Logic based on Ordinals* terminology, they sometimes seem to have access to uncomputable *oracles*. Certainly Euler and Ramanujan's extreme mathematical creativity are difficult to explain; indeed, they seem, at least superficially, to defy ordinary rational explanations.

In other words, are we machines or do we have a divine spark? In *MIND* Turing argues the former, but as Emil Post argued in the 1940s, Gödel and Turing's work on incompleteness and uncomputability can equally well be viewed as emphasizing the fundamental importance of creativity—**defined** as uncomputability—in the progress of mathematics.⁷ This also emphasizes the connection between incompleteness and uncomputability and mathematical and biological creativity, which I do not believe are that different.

In fact, in my view incompleteness and uncomputability open a door from mathematics to biology. The halting probability Ω contains infinite irreducible complexity and in a sense shows that pure mathematics is even more biological than biology itself, which merely contains extremely large finite complexity. For each bit of the dyadic expansion of Ω is one bit of irreducible mathematical information, while the human genome is merely 3×10^9 bases = 6×10^9 bits of information.

It is a delightful paradox that Turing argues that we are machines while all the while emphasizing the importance of what machines cannot do. Like a good philosopher, he cannot help seeing the good arguments on both sides. He thus provides ammunition to both parties.

⁷Post's 1941 words, first published by Martin Davis in 1965 in *The Undecidable*, and movingly placed by Jeremy Gray at the conclusion of his 2008 treatise on *Plato's Ghost: The Modernist Transformation of Mathematics*, are worth quoting: “mathematical thinking is, and must be, essentially creative.”

Part IV: The Mathematics of Emergence: The Mysteries of Morphogenesis.

Having applied mathematical methods to the foundations of reason (mathematics) and to the question of how we think, Turing obviously could not resist attempting to increase the scope of mathematical methods in biology, a subject notably resistant to mathematical reasoning.

Newton taught us to use ordinary differential equations in physics, and Maxwell taught us to use partial differential equations. Fisher-Wright-Haldane population genetics makes use of ordinary differential equations, and Turing's work on morphogenesis puts partial differential equations to good use.

Unfortunately, like many pioneers, Turing himself was caught half-way between traditional continuous mathematics and the new world revealed by *On Computable Numbers*. Newton was much more in the Middle Ages⁸ than the modern thinker he is portrayed as by Voltaire, and Darwin was less of a Darwinian extremist than many of his determined followers. In a similar manner, Turing could not, dying as he did prematurely in 1954, barely after the work of Watson and Crick in 1953, appreciate the fact that DNA is a powerful digital programming language. Indeed, DNA is presumably a universal programming language, a concept for which we are indebted to Turing.

As I have argued in my work on what I call *metabiology*, following Turing's ideas (but not his own work on biology) suggests modeling life as randomly evolving software, software that describes a random walk of increasing fitness in program space. In this manner we can discuss biological creativity, something that has gotten lost in the accounts of Darwinian evolution emphasizing competition and survival of the fittest.

Sir Ronald Fisher has been referred to as the greatest biologist of the 20th century because his population genetics gives a mathematical basis for Darwinian evolution. But by definition there is no biological creativity in population genetics, since it deals with a fixed gene pool and merely studies the changes in gene frequencies in response to selective pressures.

How then are we to understand mathematically biological creativity such as the invention of the eye or the transition from unicellular to multicellular organisms? For that it is helpful to think of DNA as randomly mutating

⁸See John Maynard Keynes' famous essay *Newton, the Man* on Newton as the last Babylonian magician.

computer software, and to study the random evolution of artificial software (computer programs) rather than natural software (DNA). In this manner it is easy to understand the absence of intermediate forms, the fact that ontogeny recapitulates phylogeny, and I have recently even been able to prove that in such a simplified setting random mutations will drive unlimited and unending biological creativity. I achieve this by forcing my organisms to work on mathematical problems for which there are no general methods and unlimited, unending creativity is essential (see my remarks on “Are we mechanical or are we creative?” in my discussion of Turing’s paper on *Computing Machinery and Intelligence* in *MIND*).

Thus my proposed new field of metabiology, which already has some mathematical successes to its credit, deliberately mixes mathematical and biological creativity in order to enable us to prove that evolution works. In my toy model of biology the *Against Method* moral of the unsolvability of the halting problem is used to drive evolution.

In fact, in one of the versions of metabiology, the organisms that rapidly evolve due to random mutations are better and better approximations to the halting probability Ω —lower bounds in fact. Indeed, all possible versions of the halting probability evolve in parallel. So random mutations yield mathematical creativity; intelligence emerges from randomness. I suspect that this version of metabiology may be the Platonic ideal of evolution that real, messy biological evolution can only approximate asymptotically in the limit from below.

In another version of metabiology I can show that hierarchical structure will rapidly emerge, which is a conspicuous feature of actual biological organisms.

How would Turing view these developments? This metabiological work on life as evolving software views organisms as machines in the spirit of Turing’s paper in *MIND*, but metabiology simultaneously takes advantage of the unsolvability of the halting problem from *On Computable Numbers* to show that evolution is unending. In a sense, we simultaneously use and refute Turing—and we “refute” him by using his own methods, so that it is clear that the contradictory spirit of Turing very much lives on in this metabiological work.⁹

⁹For more on metabiology see “Metaphysics, metamathematics and metabiology” in H. Zenil, *Randomness Through Computation*, World Scientific, in press, and “To a mathematical theory of evolution and biological creativity” at <http://www.cs.auckland.ac.nz/CDMTCS//researchreports/391greg.pdf>.