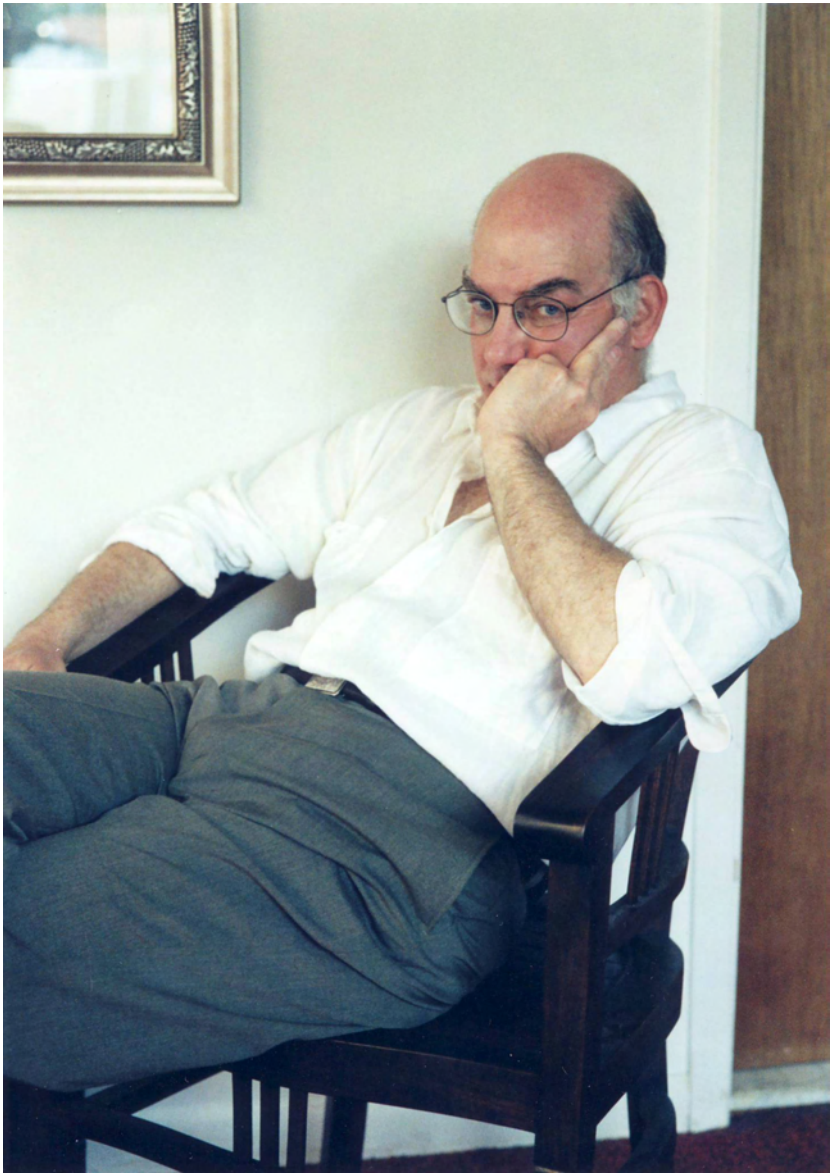
Cristian S. Calude

Randomness & Complexity, from Leibniz to Chaitin

World Scientific, Singapore, 2007



Gregory J. Chaitin (portrait by Jacquie Meyer)

Medallion presented by Stephen Wolfram to Gregory Chaitin at the NKS2007 Conference, Burlington, 15 July 2007. This is a Latin metre. The second phrase is a pentameter. With the hexameter of the first text it makes a perfect Latin elegiac couplet. The English translation is:

Everything can be summarized in one thing, but the thing itself cannot be reached: The truths of mathematics come out as fortuitous
Celebrating Gregory Chaitin MMVII

00010 00000 01000 01010 01110 11100 00111 11010 are the exact first 40 bits of Ω_{U2} calculated by Calude and Dinneen in [4]. The downward pointing arrows indicate whether or not each individual Turing machine halts.

Preface

Human beings have a future if they deserve to have a future!
Gregory J. Chaitin

This book is offered with admiration and affection to Gregory J. Chaitin[1] on the occasion of his 60th birthday (25 June 2007).

Gregory J. Chaitin, together with Ray Solomonoff and Andrei N. Kolmogorov, are the founding fathers of the subject called Algorithmic Complexity, Kolmogorov Complexity, or Algorithmic Information Theory (AIT).[2]

R. Solomonoff and G. Chaitin at NKS2003 (photo by David Reiss)

[1] Greg for friends.
[2] Chaitin coined the name AIT; this name is becoming more and more popular.

During its history of more than 40 years, AIT knew a significant variation in terminology. In particular, the main measures of complexity studied in AIT were called Solomonoff-Kolmogorov-Chaitin complexity, Kolmogorov-Chaitin complexity, Kolmogorov complexity, Chaitin complexity, algorithmic complexity, program-size complexity, etc. Solovay's handwritten notes [22][3], introduced and used the terms Chaitin complexity and Chaitin machine.[4] The book [21] promoted the name Kolmogorov complexity for both AIT and its main complexity.[5]

The main contribution shared by AIT founding fathers in the mid 1960s was the new type of complexity—which is invariant up to an additive constant—and, with it, a new way to reason about computation. Founding fathers' subsequent contributions varied considerably. Solomonoff's main interest and results are related to inductive inference, see [19]. Kolmogorov's main contributions to AIT were mainly indirect[6]—through the works of his students, P. Martin-Löf, L. Levin, V. Uspenskij.[7] Chaitin's contributions—spanning over four decades—on plain and program-size complexity, algorithmic randomness (finite and infinite sequences), applications to Gödel incompleteness, and concrete versions of AIT (hands-on programming), are central for the field. One can appreciate their lasting impact by inspecting the forthcoming monograph [17] which also includes the boom of results obtained in the last decade (due in part to the *renaissance* of Recursion Theory focussed on AIT).

While Chaitin's main contributions are in AIT, he was engaged in other research projects as well.

His first paper [5]—published when he was 18—was in automata theory. It significantly improves a theorem by Moore, which later became very important for modelling quantum phenomena with automata, see [24]. In fact, Chaitin was interested in the relation between computation and quantum physics since the early 1960s; he even wrote an APL2 course outline for

[3]During the research for my book [1] I was advised by Greg Chaitin to carefully read [22]. The problem was that the manuscript appeared to have vanished, certainly Chaitin and Solovay didn't have copies. Eventually, C. Bennett kindly sent me a copy in early 1993 and I circulated it in the community. It had the lasting effect predicted by Chaitin; see for example [17].

[4]Solovay continued to use this terminology in his later paper [23]. I used this terminology in [1].

[5]See more about the early history of AIT in [25].

[6]As Shen explained in Dagsthul, by the mid 1960s Kolmogorov's interests had more and more focused on school mathematics education.

[7]See [20] for a complete list of publications.

physics, [8].[8] His paper [6] initially included a direct reference to Nature's impact on how a Turing machine operates, but that part was removed at the referee's request. Here is how he put it recently in [3]:

> If Nature really lets us toss a coin, then, with extremely high probability, you can actually compute algorithmically irreducible strings of bits, but there's no way to do that in a deterministic world.

Chaitin has been employed by IBM for 40 years.[9] In the late 1970s and early 1980s he was part of the group designing the RISC architecture. One of his main contributions,[10] the Chaitin style graph colouring algorithm for optimal global register allocation, featured as a key innovation in the IBM 801 computer; a very influential paper [7, 9] describes this algorithm. Chaitin's intense interest in hands-on programming is also visible in his pioneering work on concrete AIT.

Chaitin has a long-time interest in philosophy, in general, and in the philosophy of mathematics, in particular. For Chaitin [11] (p. *xiii*), programming is a reliable way to achieve mathematical understanding:

> To me, you understand something only if you can program it. (You, not someone else!)

Why? Because [19] (p. 30):

> ... programming something forces you to understand it better, it forces you to really understand it, since you are explaining it **to a machine**.

He supports the view that mathematics is quasi-empirical[11] and experimental mathematics should be used more freely. In recent years his attention was captured by Leibniz, the man and the philosopher, whom he sees as precursor of the idea of descriptional complexity:

> Mais quand une regle est fort composée, ce qui luy est conforme, passe pour irrégulier.[12]

One may be led to think that philosophical interests and a focus on big ideas signal a lack of steam for technical mathematical problems. This is

[8] He is currently a member of the Physical Sciences Department at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York.
[9] An all-day celebration will be organised by the IBM Research Division on 15 November.
[10] See also [10].
[11] Chaitin's position is close to digital philosophy (E. Fredkin, S. Wolfram, and K. Zuse).
[12] But when a rule is extremely complex, that which conforms to it passes for irregular (random).

not the case as his most recent papers show [13, 14].[13]

My first contact with Chaitin's work was through Martin Davis' paper [16], which gives a beautiful presentation of one of Chaitin's information-theoretic forms of incompleteness.[14] In the late 1970's, I started a weekly research seminar on AIT at the University of Bucharest which lasted till my departure in 1992. In this seminar, we read many of Chaitin's papers and we presented some of our own; he was our main source of inspiration. I first met Greg in January 1993—he was my first visitor in Auckland, where my family relocated in December 1992. Since then, I have been very privileged to continue meeting him regularly, to understand some of his results not from papers, not from books, but from stimulating discussions, and to cooperate on different projects (including a joint paper published in *Nature*, [2], which inspired a poem [15]).

Chaitin is an unconventional person as well as an unconventional thinker and scientist. Radically new ideas are not easily accepted and more often than not generate strong opposition and resentment. One of the best "portraits" of "Chaitin in action" was painted by John Horgan in [18]:

> Stout, bald, and boyish, he wore neobeatnik attire: baggy white pants with an elastic waistband, black T-shirt adorned with a Matisse sketch, sandals over socks. He was younger than I expected; I learned later that his first paper had been published when he was only 18, in 1965. His hyperactivity made him seem younger still. His speech was invariably either accelerating, as he became carried away by his words, or decelerating, perhaps as he realized he was approaching the limits of human comprehension and ought to slow down. Plots of velocity and volume of his speech would form overlapping sine waves. Struggling to articulate an idea, he squeezed his eyes shut, and, with an agonized grimace, tipped his head forward, as if trying to dislodge the words from his sticky brain.

It would be a fatal mistake to measure the huge impact of Chaitin's *oeuvre* with a classical yardstick. We cannot understand the information compressed in a Chaitin Omega number[15], one of his major discoveries, we

[13] Chaitin's recent paper [14] was inspired by Davis' paper [16], perhaps not a random coincidence.

[14] This paper is reproduced in this book.

[15] In an article included in this book, J.-P. Delahaye observes that "the symbol Ω had been used in mathematics for a variety of purposes. But increasingly it was reserved for Chaitin's number, just as π came exclusively to represent Archimedes' constant at the beginning of the 18th century."

can get a mere[16] glimpse of it. How could we hope to understand Chaitin?

The contributions included in this book have been grouped into the following categories: technical contributions (AIT and related areas, Physics, Algebra, Automata Theory, Computer Architecture), papers on Philosophy, essays, and reminiscences. The book also includes Chaitin's own recollections on AIT, and pictures from the Chaitin celebration at the *New Kind of Science Conference* (Burlington, 15 July 2007).

I am grateful to Prof. Kok Khoo Phua, Chairman of World Scientific, for supporting this project and to Kim Tan from World Scientific for being such an efficient editor. I wish to thank Springer and the science magazine *Pour la Science* for allowing me to reprint articles initially published by them. I also thank Jeff Grote, Sally McCay, Jacquie Meyer, David Reiss, and Karl Svozil for permission to reproduce their pictures. A big thank you goes to all contributors to this volume as well as to Wolfram Research, University of Vienna, the Gödel Society and IBM Research for organising meetings in which the book was or will be presented.

Finally, to the Omega Man, to use the *Time* magazine formula, from all of us, a Very Happy Birthday!

Cristian S. Calude
Auckland, July 2007

References

[1] C. S. Calude. *Information and Randomness. An Algorithmic Perspective*, Springer-Verlag, Berlin, 1994. Second Edition, Revised and Extended, 2002.
[2] C. S. Calude, G. J. Chaitin. Randomness everywhere, *Nature* 400, 22 July (1999), 319–320.
[3] C. S. Calude, G. J. Chaitin. A dialogue on mathematics & physics, *The Rutherford Journal: The New Zealand Journal for the History and Philosophy of Science and Technology*, Vol. 2, 2006–2007, `www.rutherfordjournal.org`.
[4] C. S. Calude, M. J. Dinneen. Exact approximations of omega numbers, *Int. Journal of Bifurcation & Chaos* 17, 6 (2007), to appear.
[5] G. J. Chaitin. An improvement on a theorem by E. F. Moore, *IEEE Transactions on Electronic Computers*, EC-14 (1965), 466–467.
[6] G. J. Chaitin. On the length of programs for computing finite binary sequences, *Journal of the ACM* 13 (1966), 547–569.

[16] Finite, ridiculously small.

[7] G. J. Chaitin. Register allocation and spilling via graph coloring (with retrospective), *Best of PLDI*, 1982, 66–74, http://doi.acm.org/10.1145/989393.989403. Also in: *Proc. SIGPLAN '82 Symp. Compiler Construction*, June 1982, 98–105.

[8] G. J. Chaitin. An APL2 gallery of mathematical physics—A course outline, *Proceedings Japan 85 APL Symposium*, N:GE18–9948–0, IBM Japan, 1985, 1–56.

[9] G. J. Chaitin, M. A. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins and P. W. Markstein. Register allocation via coloring, *Computer Languages* 6, (1981), 47–57.

[10] G. J. Chaitin, C. H. Hoagland, M. J. Stephenson. System and method for generating an object module in a first format and then converting the first format into a format which is loadable into a selected computer, *United States Patent 4791558*, http://www.freepatentsonline.com/4791558.html.

[11] G. J. Chaitin. *Meta Math!*, Pantheon, New York, 2005.

[12] G. J. Chaitin. Epistemology as information theory: from Leibniz to Omega, *Collapse* 1 (2006), 27–51.

[13] G. J. Chaitin. Probability and program-size for functions, *Fundamenta Informaticae* 71 (4) (2006), 367–370.

[14] G. J. Chaitin. An algebraic characterization of the halting probability, *Fundamenta Informaticae,* 79 (1-2) (2007), 17–23.

[15] R. M. Chute. Reading a note in the journal Nature I learn, *Beloit Poetry Journal* 50 (3), Spring 2000, 8. Also in the book by the same author, *Reading Nature*, JWB, Topsham, 2006.

[16] M. Davis. What is a Computation? in L. A. Steen, *Mathematics Today: Twelve Informal Essays,* Springer-Verlag, New York, 1978, 241–267.

[17] R. Downey and D. Hirschfeld. *Algorithmic Randomness and Complexity*, Springer, Berlin, to appear in 2008.

[18] J. Horgan. *The End of Science*, Helix Books, Reading Mass., 1996, 227–228.

[19] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*, Springer, Berlin, 2004.

[20] A. N. Kolmogorov. *Selected Works of A. N. Kolmogorov. Vol. III. Information Theory and the Theory of Algorithms*, Kluwer, Dordrecht, 1993.

[21] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, Berlin, 1993. Second Edition, 1997.

[22] R. M. Solovay. *Draft of a paper (or series of papers) on Chaitin's work ... done for the most part during the period of Sept.–Dec. 1974*, unpublished manuscript, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, May 1975, 215 pp.

[23] R. M. Solovay. A version of Ω for which ZFC can not predict a single bit, in C.S. Calude, G. Păun (eds.). *Finite Versus Infinite. Contributions to an Eternal Dilemma*, Springer-Verlag, London, 2000, 323–334.

[24] K. Svozil. *Randomness & Undecidability in Physics*, World Scientific, Singapore, 1993.

[25] Panel discussion: History of Algorithmic Randomness & Complex-

ity, Dagstuhl-Seminar 06051, January 2006, http://www.hutter1.net/dagstuhl/#history. Speakers: C. S. Calude, G. Hotz, C. P. Schnorr, A. Shen, P. Vitányi.

Contents

Technical Contributions

Chapter 1

On Random and Hard-to-Describe Numbers

Charles H. Bennett[1]

IBM Watson Research Center, Yorktown Heights, NY 10598, USA;
bennetc@us.ibm.com

The first essay discusses, in nontechnical terms, the paradox implicit in defining a random integer as one without remarkable properties, and the resolution of that paradox at the cost of making randomness a property which most integers have but can't be proved to have. The second essay briefly reviews the search for randomness in the digit sequences of natural irrational numbers like π and artificial ones like Champernowne's $C = 0.12345678910111213\ldots$, and discusses at length Chaitin's definable-but-uncomputable number Ω, whose digit sequence is so random that no betting strategy could succeed against it. Other, Cabalistic properties of Ω are pointed out for the first time.

1 Berry's Paradox and the Unprovability of Randomness

The number 1,101,121 is unusual in that it is, or appears to be, the number named by the expression *the first number not nameable in under ten words.* However, since the italicized expression has only nine words, there is an inconsistency in regarding it as a name for 1,101,121 or any other number. This paradox, a variant of one due to Russell and Berry [1], shows that the concept of nameability or definability is too vague and powerful to be used without restriction. Because of it, the "function" $N(x) =$ *the number of English words required to name the integer* x must be regarded as ill-defined for all but finitely many x. Martin Gardner [2] has pointed out that a similar paradox arises when one attempts to classify numbers as "interesting" or "dull": there can be no dull numbers, because, if there were, the first of them would be interesting on that account.

[1]This paper was written and widely circulated in 1979, but is published here for the first time in its original form.

Berry's paradox can be avoided and tamed by restricting nameability to mean describability as output of an algorithm or computer program. Consider the function $C(x) =$ *the number of bits in the smallest program to compute the integer* x. A (binary) integer p is said to be a program to compute x when some standard universal computer, given p as its sole input, computes x as its sole output, afterward halting. In this case there can be no doubt that p indeed describes x. Since every integer admits such a description, $C(x)$ is well-defined for all x. However, to avoid Berry's paradox, it must be concluded that the function $C(x)$ is itself uncomputable. For if $C(x)$ were computable, one could design a contradictory program q to find and print out the least number x for which $C(x)$ exceeded the number of bits in q.

Returning to the question of interesting and dull numbers, an interesting number may without paradox be defined as one computable by a program with fewer bits than the number itself. This short description would attest some special feature of the number, by which it could be distinguished from the general run of numbers. A dull or "random" number, on the other hand, would be one that is algorithmically incompressible. Obviously, most numbers are random in this sense, since, for any n, there are more than twice as many $\leq n$-bit numbers as $(\leq n-1)$-bit numbers available to serve as shorter descriptions. Using this definition of randomness, G. Chaitin [3] demonstrated the following surprising fact, a form of Gödel's incompleteness theorem: *although most numbers are random, only finitely many of them can be proved random within a given consistent axiomatic system.* In particular, a system whose axioms and rules of inference require about n bits to describe cannot prove the randomness of any number much longer than n bits. If the system could prove randomness for a number much longer than n bits, the *first* such proof (first, that is, in an unending enumeration of all proofs obtainable by repeated application of the axioms and rules of inference) could be manipulated into a contradiction: an approximately n-bit program to find and print out the specific random number mentioned in this proof, a number whose smallest program is by definition considerably longer than n bits.

2 The Search for a "Random" Real Number

It has been conjectured that the decimal expansions of irrational numbers such as, π, e, and $\sqrt{2}$ are random in the sense of being "normal" [4] i.e. that each digit 0 through 9, and indeed each block of digits of any length,

occurs with equal asymptotic frequency. It is easy to show that no rational number is normal to any base, and that almost all irrational numbers are normal to every base; but the normality of these most famous irrational numbers remains open. The question cannot be settled by any finite amount of statistical evidence, since an ultimately normal number might begin abnormally (e.g. $e = 2.718281828\ldots$), or *vice versa.* Existing evidence [5] shows no significant departures from randomness in π. e also appears to be normal, though there is some evidence for other statistical irregularities [6].

In contrast to π, whose random-appearing digit sequence mocks the attempt to prove it so, the following very non-random number:

$$C = 0.12345678910111213141516171819202122232425262728293031\ldots$$

is nevertheless provably normal, to base 10. This number, invented by D. G. Champernowne [7], consists of the decimal integers written in increasing order (Benoit Mandelbrot has pointed out another number of this sort, whose base-2 normality is implicit in an earlier paper by N. Wiener [8]). Departures from equidistribution are large in the initial portion of Champernowne's or Wiener's number, but approach zero as the count is extended over more and more of the sequence. It is apparently not known whether these numbers are normal to every base.

Although the digit sequence of π may be random in the sense of being normal, it is definitely not random in the sense of being unpredictable: a good gambler betting against it would eventually infer its rule and thereafter always win, and only a very inept gambler could lose many bets against Champernowne's number. Is there a sequence so random that no computable betting strategy, betting against it at fair odds, can win an infinite gain? Any number that is random in this strong sense is also normal to every base. It is a basic result of probability theory that almost all real numbers are random in this strong sense [9], but here again we are seeking a *specific* random number.

There is, of course, a sense in which no specifically definable real number can be random. Since there are uncountably many real numbers but only countably many definitions, the mere fact that a real number is definable makes it atypical of real numbers in general. Here, however, we are only seeking a number whose atypicality is unrecognizable by constructive means. In particular, the number we are seeking must not be computable from its definition; since if it were, that would already imply a perfect betting strategy. One may define an uncomputable real number K in terms

of the halting problem[2] for programs on some standard universal computer or programming language, setting the n'th binary digit of K to 1 or 0 according to whether the n'th program halts. Although the resulting digit sequence is indeed uncomputable, a gambler could nevertheless make infinite profit betting against it, by betting only on solvable cases of the halting problem, of which there are infinitely many. G. Chaitin [10] discovered a real number which is uncomputable in the stronger sense needed:

Ω = *the halting probability of a universal computer whose program is generated randomly, by tossing a fair coin whenever the computer requests another bit of input.*

Clearly, once the universal computer or programming language is specified, Ω is a well defined real number between zero and one. For typical programming languages like Fortran, Ω will be nearer one than zero, since a program generated at random is more likely to halt immediately (e.g. due to a syntax error) then to loop. However, it can be shown that after the first few digits Ω would look quite random, far more than Champernowne's number.

Ω has three related properties that make it unusual:

(1) It encodes the halting problem in a very compact form. Knowing its first few thousand digits would in principle permit the solution of all interesting finitely refutable mathematical conjectures.
(2) It is algorithmically incompressible: there exists a constant c such that the first n bits of Ω are never expressible as the output of a program smaller than $n - c$ bits.
(3) No computable gambling scheme can make infinite profit betting against it.

Ω encodes the halting problem, but in a much more compact form than K: knowing its first n bits is sufficient to solve the halting problem for any

[2]The halting problem, i.e. the problem of distinguishing programs that come to a spontaneous halt from those that run on indefinitely, is the classic unsolvable problem of computability theory. At first sight the problem might seem solvable since, if a program halts, that fact can certainly be demonstrated by running the program long enough. Moreover there are many programs which can easily be proven to halt or not to halt even without running the program. The difficulty comes not in solving particular cases, but in solving the problem in general. It can be shown that there is no effective prescription for deciding how long to run a program that waits long enough to reveal the halting of all halting programs, nor any consistent system of axioms strong enough to prove the non-halting of all non-halting ones. The unsolvability of the halting problem can be derived from and indeed is equivalent to the fact that most random integers can't be proved random.

program up to n bits in length. Suppose one wishes to solve the halting problem for a particular n-bit program p. The program p corresponds to a particular sequence of n coin tosses having probability 2^{-n}, and, if it halts, contributes this amount of probability to the total halting probability Ω. Let Ω_n represent the known first n bits of Ω, so that

$$\Omega_n \quad < \quad \Omega \quad \leq \quad \Omega_n + 2^{-n}.$$

In order to decide the halting of p, begin an unending but systematic search for *all* programs that halt, of whatever length, running first one program then another for longer and longer times (cf. Fig. 1) until enough halting programs have been found to account for more than Ω_n of the total halting probability[3]. Then either p is among the programs that have halted so far, or else it will never halt, since its subsequent halting would drive the total halting probability above its known upper bound of $\Omega_n + 2^{-n}$. Note that there is apparently no way of using Ω to solve the halting problem for one n-bit program without solving the halting problem for all other $\leq n$-bit programs at the same time.

Most of the famous unproved conjectures of mathematics (Fermat's conjecture, Goldbach's conjecture, the extended Riemann hypothesis, and, until recently, the four-color problem) are conjectures of the nonexistence of something, and would be refuted by a single finite counterexample. Fermat's conjecture, for example, would be refuted by finding a solution to the equation $x^n + y^n = z^n$ in positive integers with $n > 2$; Riemann's hypothesis by finding a misplaced zero of the zeta function. Such conjectures are equivalent to the assertion that some program, which searches systematically for the allegedly nonexistent object, will never halt.

Interesting conjectures of this sort are generally sufficiently simple to describe that they can be encoded in the halting of *small* programs, a few thousands or tens of thousands of bits long. Thus only the first few thousand digits of Ω would be needed in principle to solve these outstanding "finitely refutable" conjectures as well as any others of comparable simplicity that might be thought of in the future[4].

[3]If Ω were a terminating binary rational, the expansion ending in infinitely many ones should be used, making $\Omega_n < \Omega = \Omega_n + 2^{-n}$. In fact, this problem never arises, since, as will be proved presently, Ω is irrational, and so lies strictly between Ω_n and $\Omega_n + 2^{-n}$.

[4]Some well-known conjectures, e.g. that π is normal, or that there are infinitely many twin primes (consecutive odd primes like 3 and 5 or 17 and 19), or that there are only finitely many primes of the form 2^n+1, are not in principle decidable one way or the other by any finite amount of direct evidence. Perhaps the most important conjecture of this sort is the $P \neq NP$ conjecture in computational complexity theory, which holds that there

An important class of statements decidable by Ω are statements of the form "proposition p is provable in axiomatic system $\mathbf{A}$". As indicated in the first essay, given a description of the axioms and rules of inference of $\mathbf{A}$, it is possible to effectively enumerate all possible proofs within the system, and hence all provable statements. Assuming that the proposition p and system $\mathbf{A}$ together require n bits to describe, there is a certain program of about n bits which will halt if and only if p is provable in $\mathbf{A}$. Thus, for any proposition p and axioms $\mathbf{A}$ simple enough to be "interesting", the first few thousand bits of Ω suffice to decide among the three possibilities: p is provable in $\mathbf{A}$, p is refutable in $\mathbf{A}$, or p is independent of $\mathbf{A}$. Another consequence of the enumerability of proofs, as mentioned earlier, is Chaitin's form of Gödel's theorem: even though most integers are algorithmically random, no axiomatic system describable in n bits can prove randomness for integers much larger than n bits. Ω provides a strong converse to this theorem: its first n bits constitute a sufficient "axiom" to decide the ran-

---

are problems for which the validity of a guessed solution can be tested quickly, but for which solutions cannot be found quickly. Logically, such higher level conjectures involve multiple quantifiers such as $\forall\exists$ "for infinitely many", while finitely refutable conjectures, i.e. those equivalent to the statement that a certain program will not halt, involve only a single quantifier $\forall$ "for all". Although higher level conjectures cannot be directly decided by Ω, there is good reason to believe that many of them, including most of the interesting ones, could be decided indirectly, as logical consequences of stronger, finitely refutable conjectures. For example, may twin primes are known, and empirical evidence indicates that the spacing between them grows rather slowly. Thus the twin prime conjecture may be viewed as an unnecessarily weak form of a stronger but still probably true assertion about the spacing of twin primes, say that there is always at least one pair of twin primes between 10^n and 10^{n+1}. This stronger conjecture would be decided by the early digits of Ω, since it is equivalent to the nonhalting of a simple program that looks for an excessively large gap in the distribution of twin primes. Conversely, the assertion that there are only finitely many primes of the form $2^n + 1$ may be viewed as an unnecessarily weak form of the assertion that there are fewer than, say, 10^{100} such primes, or some other easily-named large number (in fact only six are known). Like the strengthened form of the twin prime conjecture, this assertion is equivalent to the non-halting of a simple program, one that looks for the 10^{100}'th prime of the specified form. Similarly, the normality of π and the inequality of P and NP would follow from stronger, finitely refutable statements supported by the same evidence as the original conjectures. In all these cases the finitely refutable statement is obtained by assuming a generous but computable bound on one of the original statement's existential quantifiers. Aside from these conjectures, which probably follow from finitely refutable ones, there are some mathematical statements that definitely cannot be reduced to halting problems. Typical of these are some statements about Ω itself, e.g. "the j'th bit of Ω is a 1". By virtue of the incompressibility of Ω, the first n members of this family of two-quantifier statements [10] cannot be decided by any algorithm smaller than about n bits. This implies, incidentally, that no irrational number can efficiently encode the decision of all higher level statements the way Ω encodes the decision of all finitely refutable ones.

domness of all integers of $n+1$ bits or less. The procedure for doing this is essentially the same as that used to solve the halting problem: to find whether a given $n+1$ bit integer x is algorithmically random, use Ω_n as described earlier to find all $\leq n$-bit programs that halt. If none of these has x as its output, then by definition x is algorithmically random.

Let us now return to the senses in which Ω itself is random: its incompressibility and the impossibility of successfully gambling against it. It may appear strange that Ω can contain so much information about the halting problem and yet be computationally indistinguishable from a meaningless random sequence generated by tossing a coin. In fact, Ω is a totally informative message, a message which appears random because all redundancy has been squeezed out of it, a message which tells us only things we don't already know.

To show that Ω is incompressible, let p be a program that for some n computes Ω_n, the first n bits of Ω. This program may be altered, increasing its size by at most c bits (c a constant independent of n), so that instead of printing Ω_n it finds and prints out the first algorithmically random $(n+1)$-bit number, as explained above. This would be a contradiction unless the original program p were at least $n-c$ bits long.

No finitely describable computable gambling scheme can win an infinite profit betting against the bits of Ω. Let G be a gambling scheme, describable in g bits, and able to multiply the gambler's initial capital 2^k fold by betting on some number n of initial bits of Ω. Without loss of generality we may suppose that the scheme includes a specification of the desired gain 2^k, and that it quits as soon as this gain is achieved, making no further bets. One may imagine the same gambling scheme applied to other inputs besides Ω. On most of them it would fail to achieve its goal, but on some it would succeed. Indeed one may use G to enumerate *all* the finite inputs on which G would quit successfully. This set has total probability 2^{-k} or less, of which 2^{-n} is contributed by Ω_n.

It can be shown [10] that about $n-k$ bits suffice to locate Ω_n within this enumeration of successful inputs. Therefore Ω_n can be computed by a program of approximately $g+n-k$ bits. This means, in turn, that k cannot be much greater than g without violating the incompressibility of Ω. Therefore no g-bit gambling scheme betting on Ω can multiply the initial capital by more than about 2^g, the amount one would win by simply knowing g bits of Ω and betting only on those bits.

Throughout history philosophers and mystics have sought a compact key to universal wisdom, a finite formula or text which, when known and

understood, would provide the answer to every question. The Bible, the Koran, the mythical secret books of Hermes Trismegistus, and the medieval Jewish Cabala have been so regarded. Sources of universal wisdom are traditionally protected from casual use by being hard to find, hard to understand when found, and dangerous to use, tending to answer more and deeper questions than the user wishes to ask. Like God the esoteric book is simple yet undescribable, omniscient, and transforms all who know It. The use of classical texts to foretell mundane events is considered superstitious nowadays, yet, in another sense, science is in quest of its own Cabala, a concise set of natural laws which would explain all phenomena. In mathematics, where no set of axioms can hope to prove all true statements, the goal might be a concise axiomatization of all "interesting" true statements.

Ω is in many senses a Cabalistic number. It can be known of, but not known, through human reason. To know it in detail, one would have to accept its uncomputable digit sequence on faith, like words of a sacred text. It embodies an enormous amount of wisdom in a very small space, inasmuch as its first few thousand digits, which could be written on a small piece of paper, contain the answers to more mathematical questions than could be written down in the entire universe, including all interesting finitely-refutable conjectures. Its wisdom is useless precisely *because* it is universal: the only known way of extracting from Ω the solution to one halting problem, say the Fermat conjecture, is by embarking on a vast computation that would at the same time yield solutions to all other equally simply-stated halting problems, a computation far too large to be carried out in practice. Ironically, although Ω cannot be computed, it might accidentally be generated by a random process, e.g. a series of coin tosses, or an avalanche that left its digits spelled out in the pattern of boulders on a mountainside. The initial few digits of Ω are thus probably already recorded somewhere in the universe. Unfortunately, no mortal discoverer of this treasure could verify its authenticity or make practical use of it.

The author has received reliable information, from a Source who wishes to remain anonymous, that the decimal expansion of Ω begins

$$\Omega \quad = \quad 0.9999998020554253273471801908\ldots$$

References

(1) Whitehead, A. N., and Russell, B., *Principia Mathematica,* Vol. **1**, Cambridge University Press, London (1925) p. 61.

(2) Gardner, M., *Sci. Amer.* **198** (1958) No. 1, p. 92.
(3) Chaitin, G., *J. Assoc. Comput. Mach.* **21** (1974) p. 403;
Sci. Amer. **232** (1975) No. 5, p. 47.
(4) Borel, E., *Rend. Circ. Mat. Palermo* **27** (1909) p. 247.
(5) Pathria, R. K., *Mathematics of Computation* **16** (1962) p. 188.
(6) Stoneham, R. G., *Amer. Math. Monthly* **72** (1965) p. 483.
(7) Champernowne, D. G., *J. Lond. Math. Soc.* **8** (1933) p. 254.
(8) Wiener, N., *Acta Math.* **55** (1930) p. 117, eq. 11.03.
(9) Martin-Löf, P., *Information and Control* **9** (1966) p. 602.
Schnorr, C. P., *Math. Systems Theory* **5** (1971) p. 246.
(10) Chaitin, G., *J. Assoc. Comput. Mach.* **22** (1975) p. 329.

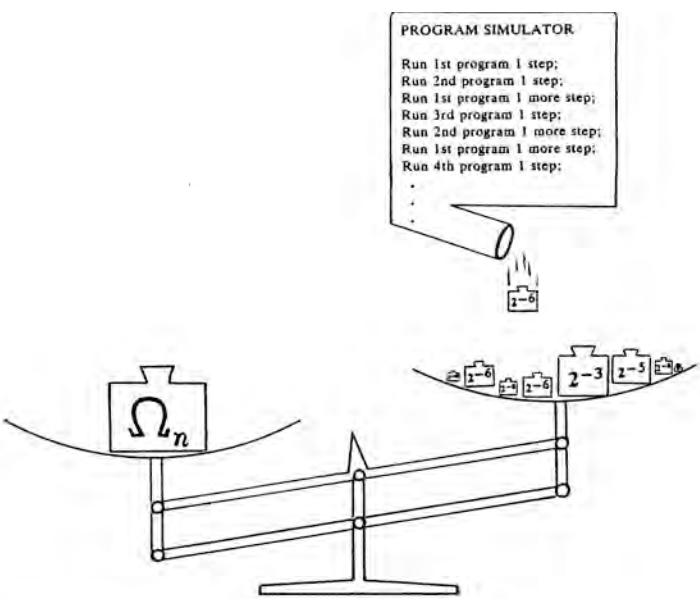


Fig. 1. Method of using Ω to solve the halting problem for all programs of length n. Let the first n digits of Ω be known; call them Ω_n. Place a weight equal to Ω_n in the left pan of a balance. Meanwhile begin a systematic but unending search for programs that halt, running one program then another for greater and greater time in the manner of the song "The Twelve Days of Christmas". Every time a program of length k is found to halt, having caused the computer to read neither more nor less than its full k bits in the course of the computation, add a weight 2^{-k} to the right pan of the balance,

because 2^{-k} is this program's probability of being chosen and executed by a computer whose input bits are supplied by a coin tossing. Eventually the balance must tip to the right, since the total weight of programs that halt, i.e. the halting probability Ω, is an irrational number between Ω_n and $\Omega_n + 2^{-n}$. After the balance has tipped, no more programs of length $\leq n$ bits can halt, because, if one did, that would raise Ω above its known upper bound of $\Omega_n + 2^{-n}$.

Chapter 2

Computing Beyond Classical Logic: SVD Computation in Nonassociative Dickson Algebras

Françoise Chaitin-Chatelin

Université Toulouse 1 and CERFACS,42 avenue G. Coriolis 31057 Toulouse Cedex 1, France; E-mail: chatelin@cerfacs.fr

This short note puts the fundamental work of G. Chaitin into an historical perspective about the multisecular evolution of the art of computing. It recalls that each major step forward in the creation of new numbers was met by strong opposition. It shows, by way of an example taken from SVD computation in nonassociative Dickson algebras, why classical logic cannot account for certain results which carry global information.

2.1. Introduction

The scientific œuvre of Gregory Chaitin revolves around computation and displays a remarkable unity of thought. More than 4 decades ago, Chaitin began to explore the limits of computation within the paradigm of a Turing machine. This led him to the celebrated Omega number [1,2] which expresses the ultimate in uncomputability à la Turing.

The Turing "thesis" about computability is an axiomatic definition of what can be computed (by a machine) within the limits of classical logic, the rational logic based on elementary arithmetic. This axiom is now accepted by computer scientists and logicians as a universal rule for computation. Therefore the work of Chaitin, which questions this claim to universality from *within*, has aroused passionate and antagonistic reactions, positive and negative.

One of the main reasons for the irrational passion stirred by his work is that it is rooted at a most fundamental level. Few questions reach deeper into human understanding than "What can rational computation achieve?".

From the point of view of classical logicians, the theoretical findings of Chaitin about computation are unacceptable. Not because the mathematics are wrong—they are impeccable—, but because the conclusions are

viewed as heretical. Some of these logicians have expressed their criticisms in a forceful way [12]. However, the absolute faith that computer scientists put in the universal validity of the axiom of Turing is not equally shared by everyone in the scientific community. Highly successful books by Penrose [10, 11] and Wolfram [13] testify to the necessity to explore other computational routes. An extension of the classical logic based on quantum coherence was already advocated by D. Deutsch in the 1980's [9]. This quantum logic led to the development of quantum computing. Experiments have shown that such a computation is physically realizable at the atomic level (Zeilinger).

There are converging indications that new kinds of logic are required to understand the real world which extends around and inside us. This has not, however, mellowed the criticisms raised by orthodox logicians against the pioneering insights of Gregory Chaitin.

This is not at all surprising. There are many historical cases of the rejection, by the vast majority of mathematicians, of radically new ideas, which, much later, were recognised as fundamental to the advance of mathematical understanding [7, 8]. Among the best known examples, one finds *new kinds of numbers:* i) *negative* numbers (12^{th}-15^{th} Century), ii) *complex* numbers (16^{th}-19^{th} Century), iii) *quaternions* (1843). Before being finally incorporated into the mathematical corpus, each new kind of numbers was met by scepticism at best, and its significance was passionately debated [3, 4, 7, 8].

This is all too understandable: each new number contradicted a commonly shared opinion of the time, implicitly taken as a universal law of computation. These opinions were respectively the following:

i) all equations have either positive solutions, or no solution (Middle Ages),
ii) any nonzero number has a positive square (late Renaissance),
iii) the multiplication of numbers is commutative (early 19^{th} Century).

The discovery of each of these new numbers was a major step forward in the evolution of the art of computing in the western world. This advance, which spanned over seven centuries, was instrumental in the axiomatic clarification of the foundations of mathematics which occured at the dawn of the 20^{th} Century. Thereafter, even associativity became an optional feature for multiplication.

2.2. Nonassociativity of multiplication

The notion of associativity was invented by Hamilton in July 1844 when he realised that the multiplication of two octonions was *not* associative. The *octonions* had been discovered 6 months earlier by his classmate J. T. Graves, a lawyer at the University of London, in an effort to derive an 8 squares theorem on the model of Hamilton's quaternions. Such a discovery was extremely ahead of its time. The non-commutative quaternions were then hardly accepted by mathematicians. Their use was to be aggressively questioned by eminent American physicists (Gibbs-Heaviside) [8].

Despite this opposition, the *associative* algebras of Clifford (1878), which extend the quaternions, have been successful tools for the development of algebraic geometry and theoretical physics until to-day [4].

Two kinds of nonassociative algebras participated in the success: the algebras of Lie and of Pascual Jordan. This very success did cast a shadow on the role of other nonassociative algebras, such as Graves' octonions, in the analysis of computation. The 8D-octonions are the smallest of the nonassociative Dickson algebras.

2.3. Nonassociative Dickson algebras

2.3.1. *Presentation of Dickson's doubling process*

The three associative Dickson algebras A_k, $k = 0$ to 2, define successively the reals, $A_0 = \mathbb{R}$, the complexes $A_1 = \mathbb{C}$, and the quaternions $A_2 = \mathbb{H}$.

The nonassociative algebras A_k extend, for $k \geq 3$, the quaternions in a way different from Clifford's. Multiplication and conjugation are inductively defined so that

$$1_{k+1} = (1_k, 0), \quad \tilde{1}_{k+1} = (0, 1_k)$$

$$A_{k+1} = A_k \times 1_{k+1} \oplus A_k \times \tilde{1}_{k+1}, \quad k \geq 0,$$

where 1_k is the real unit of A_k [4].

This inductive process defines, from $A_0 = \mathbb{R}$ at the beginning, an endless chain of *complexified* algebras A_k of dimension 2^k, $k \in \mathbb{N}^*$.

The process was observed by Dickson around 1912, and presented for $k = 2$ as a computational way to induce the multiplication table for the octonions, which had been given independently by Graves (1844) and Cayley (1845), from Hamilton's multiplication table for the quaternions (1843).

It is conventional wisdom that the lack of associativity is a severe limitation for computation in A_k, $k \geq 3$. Nothing could be further from reality, as this was shown in [3, 4]. Nonassociativity creates computational *opportunities* which are well exemplified by the Singular Value Decomposition (SVD) for the left multiplication map $L_a : x \mapsto a \times x,\ x \in A_k$ (Section 4).

Let $[x, y, z] = (x \times y) \times z - x \times (y \times z)$ denote the *associator* for x, y, z in A_k, $k \geq 3$.

Remark. Vectors in Dickson algebras have been called hypercomplex numbers in the 19^{th} Century. And computation on hypercomplex numbers is classically known as "hypercomputation" [3,4,7]. This mathematical notion should not be confused with a recent version of computation designed by computer scientists to overcome some of Turing's limitations (see `http://en.wikipedia.org/wiki/Hypercomputation`.)

2.3.2. *Alternative vectors in A_k, $k \geq 4$*

An alternative vector a in A_k satisfies the weakened associativity condition:

$$-[a, a, x] = ||a||^2 x + a \times (a \times x) = 0$$

for any x in A_k. The condition is identically satisfied for $k \leq 3$, but not for $k \geq 4$. All canonical basis vectors $e_i,\ i = 0$ to $2^k - 1$, are alternative for arbitrary k. Among them, the two vectors $1 = e_0$ and $\tilde{1} = e_{2^{k-1}}$ have stronger properties. They span the subalgebra $\mathbb{C}_{\tilde{1}} = \text{lin}(1, \tilde{1})$ isomorphic to $\mathbb{C}$. Any pair of vectors (x, y) in $\mathbb{C}_{\tilde{1}}$ satisfies

$$[x, x, y] = [x, y, y] = 0.$$

The vectors in $\mathbb{C}_{\tilde{1}}$ are *fully* alternative for $k \geq 4$ [3].

2.3.3. *The splitting $A_k = \mathbb{C}_{\tilde{1}} \oplus \mathcal{D}_k$, $k \geq 2$*

Let a be in A_k. It can be represented as the sum

$$a = \alpha + \beta\tilde{1} + c$$

where $h = \alpha + \beta\tilde{1} \in \mathbb{C}_{\tilde{1}}$ is the fully alternative *head* and c is the *tail*: c belongs to the subspace $\mathcal{D}_k = \mathbb{C}_{\tilde{1}}^{\perp}$ of vectors with zero component on 1 and on $\tilde{1} = \tilde{1}_k$. These vectors are called "doubly pure". Such a splitting plays an important role in non classical SVD calculations in A_k, $k \geq 3$ (Section 4).

2.4. SVD computation in $\mathcal{D}_k$ and A_k, $k \geq 3$

The notion of singular values for a matrix (or a linear map) plays an essential role in matrix computations, in particular for backward analysis when the data are uncertain [7]. It dates back to Camille Jordan (1873).

For $a \in A_k$, the singular values of the map L_a are the non-negative square roots of the eigenvalues of the symmetric map $L_a^T L_a$. For $k \leq 3$, $\overline{a} \times (a \times x) = ||a||^2 x$ for any x: there is a *unique* singular value $||a||$ for L_a, and $||a|| = 0$ iff $a = 0$. But this need not be true anymore for $k \geq 4$, unless a is alternative.

2.4.1. $c \in \mathcal{D}_k$ *is doubly pure,* $k \geq 4$.

Let c be in $\mathcal{D}_k$, $k \geq 4$. There are between 1 and 2^{k-2} distinct nonnegative singular values for L_c for $k \geq 5$. For $k = 4$, the number reduces to 1 or 3. The multiplicities are multiples of 4. The euclidean norm $||c||$ is always one of the singular values, corresponding to the 4D-singular subspace $\mathbb{H}_c$ (isomorphic to $\mathbb{H}$) spanned by $1, c, \tilde{c}$ and $\tilde{1}$, with $\tilde{c} = c \times \tilde{1}$ [3].

c is a zero divisor iff $\mathrm{Ker} L_c \neq \{0\}$, that is iff 0 is one of the singular values for L_c. It can be proved [3] that a zero divisor is necessarily doubly pure. When c is not a zero divisor, c^{-1} is uniquely defined by $c^{-1} = -c||c||^{-2}$. Therefore $L_c^{-1} \neq L_{c^{-1}} = -\dfrac{1}{||c||^2} L_c$, and $\dfrac{\|L_c\|}{\|c\|} = \|c\| \|L_{c^{-1}}\|$ represents the largest normalized singular value.

2.4.2. *Deriving the SVD of a in* A_k *from that of the tail c in* $\mathcal{D}_k$, *for* $k \geq 4$

Let c be given in $\mathcal{D}_k$ such that $||c|| = 1$. The spectrum of $-L_c^2$ is denoted $\sigma_c = \sigma(-L_c^2)$, and λ is any eigenvalue in σ_c, $\lambda \geq 0$. Let $\lambda = 1^4$ denote the eigenvalue associated with the 4D-eigenspace $\mathbb{H}_c$. The notation $\lambda \neq 1^4$ means that either $\lambda \neq 1$, or, if $\lambda = 1$, its multiplicity is ≥ 8. We set $N_\lambda = \alpha^2 + \beta^2 + \lambda$, for $\lambda \in \sigma_c$, thus $N_\lambda \geq ||h||^2 > 0$ for $h \neq 0$.

Theorem 2.1. *For* $a = \alpha + \beta\tilde{1} + c$, $c \in \mathcal{D}_k$, $L_a^T L_a = (\alpha^2 + \beta^2)I - L_c^2$. *The eigenvalues of* $L_a^T L_a$ *are* N_λ, *for* $\lambda \in \sigma_c$ *with the same multiplicities.*

Proof. Direct computation of $(\alpha - \beta\tilde{1} - c) \times (\alpha x + \beta\tilde{1} \times x + c \times x)$, $x \in A_k$. One checks that $c \times (\tilde{1} \times x) + \tilde{1} \times (c \times x) = 0$ for $x \in \mathbb{H}_c$ and $x \in \mathbb{H}_c^{\perp}$. The conclusion follows. □

The *classical* derivation of the SVD for L_a from that for L_c yields a generalization of Pythagoras theorem to $||h|| \neq 0$ and to the singular values for $L_c,\ c \neq 0$:

$$a = h + c \Longrightarrow N_\lambda = ||h||^2 + \lambda > 0 \text{ for } h \neq 0.$$

We have discovered (2005) that the nonassociative nature of multiplication in Dickson algebras for $k \geq 3$ enables us to perform a *non classical* derivation, which is a computational artifact in A_k, $k \geq 3$ [3].

2.4.3. *Nonclassical derivation from c to a, $k \geq 3$*

The nonclassical mode of derivation is defined in [3,Section 9]. It uses the block-diagonal form of $L_a^T L_a$ (with blocks of order 4) written in the eigenbasis for $-L_c^2$. In this nonclassical approach, the order in which the addition of α and $\beta\tilde{1}$ is performed *matters*. From an SVD point of view, addition is not always associative in A_k, $k \geq 3$, as we shall see.

When $\alpha\beta \neq 0$, there are 3 different routes to go from c to a in $\mathbb{C}_{\tilde{1}}$, as sketched on Figure 1: one can reach a either directly (diagonally) or sideways through $d = \beta\tilde{1} + c$, or through $e = \alpha + c$. When $\alpha\beta = 0$, the route is unique.

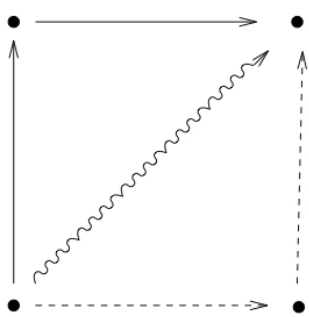


Figure 2.1. Three routes from c to a in $\mathbb{C}_{\tilde{1}}$ for $\alpha\beta \neq 0$

Section 9 in [3] uses (implicitly) the direct route $a = h + c$, which yields the same results as the sided one through $e\ :\ a = \beta\tilde{1} + e$. The two routes through d and e give different results for $\alpha\beta \neq 0$.

We define $\xi\ :\ (s,t) \in \mathbb{R} \times \mathbb{R}^+ \mapsto \xi(s,t) = ((s-t)^2, (s+t)^2) \in (\mathbb{R}^+)^2$

where $t \geq 0$, $s \in \mathbb{R}$.

Theorem 2.2. *For $a = \alpha + \beta\tilde{1} + c$, $c \in \mathcal{D}_k$, $k \geq 3$, the nonclassical SVD derivation yields the nonnegative values listed below in two columns:*

λ	*via d*	*direct or via e*
1^4	$N_1 = \alpha^2 + \beta^2 + 1$	N_1
$0 < \lambda \neq 1^4$	$N_\lambda \pm 2\beta\sqrt{\lambda}$ $= \alpha^2 + \xi(\beta, \sqrt{\lambda})$	$N_\lambda \pm 2\beta\sqrt{\lambda + \alpha^2}$ $= \xi(\beta, \sqrt{\lambda + \alpha^2})$
0	$N_0 = \alpha^2 + \beta^2$	$N_0 \pm 2\beta\alpha = (\alpha \pm \beta)^2$

Table 1

Proof. Based on [3, Lemma 9.3 and Proposition 9.4]. □

For each λ in σ_c and for $\alpha\beta \neq 0$, there are $1, 3$ or 4 different singular values when they are computed non classically in A_k, $k \geq 3$. All results for $0 < \lambda \neq 1^4$ differ from the exact value N_λ given in Theorem 4.1 when $\beta \neq 0$, with common mean. We now take a fresh look at the logical paradox arising from the existence of split zero divisors for $\beta^2 = \lambda + \alpha^2$ [3].

2.5. Is the nonclassical SVD derivation absurd?

2.5.1. *The conventional analysis*

From the point of view of classical logic, the nonclassical SVD results are plainly **wrong**, since for $0 < \lambda \neq 1^4$ they do not agree with the exact value N_λ. Moreover, when $\beta^2 = \alpha^2 + \lambda$, they contradict the theoretical result that zero divisors necessarily belong to $\mathcal{D}_k$. At face value, nonclassical SVD seems **absurd**, and it should be rejected by any sane mathematician. Or should it not?

Should we think twice? In the 16^{th} Century, $\sqrt{-1}$ was a complete mystery, which appeared totally absurd at first sight. It took three centuries of painful reflections to master its meaning as the "imaginary" unit i. Once tamed, $i = \sqrt{-1}$ found its way in almost all engineering calculations of the 19^{th} Century which dealt with wave propagation (light, sound, electricity,magnetism,...).

Warned by history, we should be extremely cautious. We should not jump hastily to the "obvious" conclusion. Could it be possible that nonclassical SVD computation *serves a purpose* from a computational point of view, and that it delivers useful *information*?

2.5.2. *Induction and nonclassical singular values*

To the vector $a = \alpha + \beta\tilde{1} + c$ in A_k, we associate $\varphi = (\alpha + c, \beta\tilde{1})$ in A_{k+1}, for $k \geq 3$. We still assume that $||c|| = 1$, $c \in \mathcal{D}_k$, and $\beta \neq 0$. Observe that $||\varphi|| = ||a|| = \sqrt{N_1}$.

Theorem 2.3. *The eigenvalues of $L_\varphi^T L_\varphi$ are given in Table 1 by the left most column. Their values equal the nonclassical eigenvalues for $L_a^T L_a$, computed in A_k via $d = \beta\tilde{1} + c$. Their multiplicities are multiplied by 2.*

Proof. Let $\varphi = (\alpha + c, \beta\tilde{1})$, and $v = (x, y)$. Direct computation of $\overline{\varphi}(\varphi \times v)$ shows that $L_\varphi^T L_\varphi$ has the 2×2 block representation

$$L = \left(\begin{array}{c|c} M & G \\ \hline -G & M \end{array}\right)$$

where $M = (\alpha^2 + \beta^2)I - L_c^2 = L_a^T L_a$, $G = -G^T = \beta[c, -, \tilde{1}]$. Now $Gx = \beta[c, x, \tilde{1}]$ is 0 for $x \in \mathbb{H}_c$ by associativity, and equals $2\beta\widetilde{c \times x}$ for $x \in \mathbb{H}_c^\perp$.

It is easily shown that L has a block diagonal structure, with blocks of order 8, derived from the eigenstructure for $-L_c^2$, see [3,Section 11].

For $\lambda = 1^4$ or 0, the corresponding blocks $N_\lambda I_8$ are diagonal. For $0 < \lambda \neq 1^4$, the blocks are of the form $N_\lambda I_8 + 2\beta\sqrt{\lambda}J$, with $J = \left(\begin{array}{c|c} 0 & K \\ \hline -K & 0 \end{array}\right)$ and $K = \left(\begin{array}{c|c} 0 & -I_2 \\ \hline I_2 & 0 \end{array}\right)$. K is antisymmetric, $K^T = -K$, $K^T K = -K^2 = I_4$. Its eigenvalues are $\pm i$ and its singular values are 1 quadruple. Thus the eigenvalues of J are given by the quadruple pair ± 1. And the eigenvalues of $L_\varphi^T L_\varphi$ are $N_\lambda \pm 2\beta\sqrt{\lambda}$ for $0 < \lambda \neq 1^4$. □

We have been able to interpret half of the seemingly meaningless singular values in A_k by the singular values of $(a + c, \beta\tilde{1})$ in the complexified algebra $A_{k+1} = A_k \oplus A_k \times \tilde{1}_{k+1}$.

This is not a complete surprise. The interpretation of the nonclassical singular values by induction from A_k to A_{k+1} mimics, for $k \geq 3$, the interpretation of $\sqrt{-1}$ from $\mathbb{R}$ to $\mathbb{C}$ ($k = 0$). What seemed at first impossible or absurd at a given level (dimension 2^k) can be resolved and understood easily at the next level (dimension 2^{k+1}).

However, this is just the tip of the iceberg, since any a in A_k can induce 4 or 8 different vectors in A_{k+1}. A more complete study can be found in [6]. It sheds light on the role of nonclassical SVD in the process of **creation** by hypercomputation.

2.6. Conclusion

The moral of this story about computation with hypercomplex numbers has already been given by Leibniz more than 300 years ago: "There is hardly any paradox without its proper role". And history tells us that extreme caution should be used before judging, based on past experience, that certain computations are absurd or impossible. Computation in nonassociative Dickson algebras begs for an extension of classical logic. It calls for a dynamical logic where the results of a computation can be right *and* wrong, depending on the *point of view*.

For example, in A_k, $k \geq 3$, $d = \beta\tilde{1} + c$ in $\mathcal{I}m\ A_k$ is alternative iff c is alternative in $\mathcal{D}_k$. Thus d cannot be a zero divisor in A_k when we assume c to be alternative. For $|\beta| = ||c||$, $\varphi = (c, \beta\tilde{1})$ is a zero divisor in $\mathcal{D}_{k+1}$ [3]. This property is indicated by the *nonclassical* singular values: one is 0, the other is $2||c||$. These 2 values are **wrong** in relation with a, in A_k, but they are the **exact** singular values for L_φ in $\mathcal{D}_{k+1}$. The exact classical singular value relative to a is, of course, $\sqrt{2}||c|| = ||a|| = ||\varphi||$, but it is mute about the 2 other singular values for L_φ.

This internal dynamical relativity of viewpoints created by *induction* exists for each level k. The limit as $k \to \infty$ defines an evolution which is clearly beyond the reach of any Turing machine [6].

If one wants to understand the manifested world, the moving, flexible world that one sees and experiences, it is necessary to scrutinise the way *information* is being dynamically processed during computation. This necessity was sensed by Gregory Chaitin already in the mid 1960's when he conceived of his Algorithmic Information Theory (AIT). His theory explores the limits of formal axiomatic reasoning based on the Turing paradigm. As was mentioned in the introduction, Chaitin exposes the limitations from *within* the paradigm. It is clear that Dickson's hypercomputation lies *outside* the paradigm, shedding a complementary light on the limitations from *without*.

Time will come when it will be obvious that the Turing thesis is a straight-jacket imposed on computation to make it mechanical. Time will come when the message of Chaitin about the limitations of purely rational computation and of axiomatic reasoning will be received by everyone [2].

There are many ways out of the evolutive dead-end that would result from any axiomatically constrained computation, such as the one that was imagined in the 20^{th} Century by Hilbert (1900) and Turing (1936).

A few such examples were mentioned in the introduction. We presented in some detail another example set in the framework of nonassociative Dickson algebras, for which an extension of classical logic beyond Turing is meaningful from the point of view of information: it takes into account the duality of viewpoints based on induction. Computation in Dickson algebras defines its own internal dynamics for evolution by successive complexification. The internal complexity differs from, yet is complementary to, the descriptive complexity of AIT. In AIT, one considers the complexity from the viewpoint of an observer who *simulates* the phenomenon by running a program, but is not a player in the evolution.

Nonassociative Dickson algebras appear as a natural framework for nonlinear computation of the kind provided by Life itself. Hypercomputation helps us understand some of Life's computing mechanisms which are not revealed by associativity.

Even more than physics, biology, and Life sciences in general, are in desperate need for new computational logics. Logics which can explain how information is being processed by living organisms during their evolution. Chaitin is one of the forerunners in this quest.

Acknowledgement

The author wishes to thank Gregory Chaitin for countless illuminating discussions over the past 14 years. The views expressed above about the Turing thesis on computability are entirely hers.

References

[1] G. Chaitin (2005) *Meta Math! The Quest for Omega,* Pantheon Books, New York

[2] G. Chaitin (2006) *The limits of reason,* Scientific American **294**, 74-81, March 2006.

[3] F. Chaitin-Chatelin (2005). *Inductive multiplication in Dickson algebras*, Technical Report TR/PA/05/56, CERFACS, Toulouse, France, 2005.

[4] F. Chaitin-Chatelin (2006). *Calcul non linéaire dans les algèbres de Dickson*, Technical Report TR/PA/06/07, CERFACS, Toulouse, France, 2006.

[5] F. Chaitin-Chatelin (2006). *Computation in nonassociative Dickson algebras*, Invited talk, July 2006, Symposium in honour of Prof. Jacqueline Fleckinger, Univ. Toulouse 1.

[6] F. Chaitin-Chatelin (2007). *About an organic logic ruling the continuous*

evolution of SVD measurement with Dickson hypercomplex numbers, Cerfacs Technical Report TR/PA/07/55.[1]

[7] F. Chaitin-Chatelin, E. Traviésas-Cassan (2005) *Qualitative computing*, Chapter 5 in *Accuracy and Reliability in Scientific Computing* (Bo Einarsson ed.), p.77-92, SIAM Philadelphia. Also Cerfacs Report TR/PA/02/58.

[8] M.J. Crowe (1994) *History of Vector Analysis,* Dover Publ. Inc., New York.

[9] D. Deutsch (2000) *The Fabric of Reality,* Penguin, London.

[10] R. Penrose (1994) *Shadows of the Mind: A Search for the Missing Science of Consciousness,* Oxford University Press, New York.

[11] R. Penrose (2004) *The Road to Reality: A Complete Guide to the Laws of the Universe,* Jonathan Cape, London

[12] P. Raatikainen (2001) Review of *The Unknowable* and *Exploring Randomness* (by G. Chaitin), Notices of the AMS **48**, 992-996.

[13] S. Wolfram (2002) *A New Kind of Science,* Wolfram Media, Champaign, IL

[1] All Cerfacs Reports are available from `http://www.cerfacs.fr/algor/reports/index.html`

Chapter 3

Janus–Faced Physics: On Hilbert's 6th Problem

N. C. A. da Costa[α1], F. A. Doria[β]

[α] *Institute for Advanced Studies, University of São Paulo, Av. Prof. Luciano Gualberto, trav. J, 374, 05655–010 São Paulo, SP Brazil*
Academia Brasileira de Filosofia, R. do Riachuelo, 303 20230–011 Rio de Janeiro, RJ Brazil; fadoria@gmail.com

[β] *Fuzzy Sets Lab, Production Engineering Program, COPPE–UFRJ, P.O.Box 68507, 21945–972 Rio, RJ Brazil;* fadoria2001@yahoo.com.br

If geometry is to serve as a model for the treatment of physical axioms, we shall try first by a small number of axioms to include as large a class as possible of physical phenomena, and then by adjoining new axioms to arrive gradually at the more special theories. ... The mathematician will have also to take account not only of those theories coming near to reality, but also, as in geometry, of all logically possible theories. We must be always alert to obtain a complete survey of all conclusions derivable from the system of axioms assumed.

D. Hilbert, 1900.

3.1. Prologue

We argue that physics can be seen as a Janus–faced discipline whose theories may be conceived as a kind of computational device, as suggested by Chaitin, which is then complemented by a conceptual play in the sense that we elaborate here. We take our cue from Hilbert's 6th Problem (the axiomatization of physics) and present an axiomatic unified treatment for classical physics (classical mechanics, first–quantized quantum mechanics, electromagnetism, general relativity and gauge field theory) based on what we call Suppes predicates. We then obtain several undecidability and in-

[1]Corresponding author

completeness results for the axiomatic systems which are developed out of a simple set of concepts where we have cradled significant portions of physics and show that they imply the undecidability of many interesting questions in physics and the incompleteness of the corresponding formalized theories. We give chaos theory as an example. Those results point towards the conceptual depth of the 'game of physics.' Our presentation follows several results that we have obtained in the last two decades.

Why axiomatize a scientific theory? Doesn't the extra rigor required carry with itself an unwanted burden that hinders the understanding of the theory's concepts?

We axiomatize a theory not only to better understand its inner workings but also in order to obtain metatheorems about that theory. We will therefore be interested in, say, proving that a given axiomatic treatment for some physical theory is incomplete (that is, the system exhibits the incompleteness phenomenon), among other things. As a follow–up, we would also like to obtain examples, if any, of physically meaningful statements within that theory that are formally independent of its axioms.

Out of the authors' previous work [15–18] we describe here a technique that allows for such an axiomatization. Its main guidelines are:

- First, the mathematical setting of the theory is clarified, and everything is formulated within the usual standards of mathematical rigor.
- We then formulate those results within an adequate axiomatic framework, according to the prescriptions we present in this paper.

We may also be interested in semantic constructions. As the required step to obtain those results, we show here how to embed a significant portion of classical physics within a standard axiomatic set theory such as the Zermelo–Fraenkel system together with the Axiom of Choice (ZFC set theory). By classical physics we mean classical mechanics as seen through the analytico–canonical (Lagrangian and Hamiltonian) formalism; electromagnetic theory; Dirac's theory of the electron — the Schrödinger theory of the electron is obtained through a limiting procedure; general relativity; classical field theories and gauge field theories in particular.

Then, it is possible to examine different models for that axiom system and to look for sentences that are true or not depending on their interpretation — and that hopefully have corresponding different physical interpretations.

It is obvious that the crucial idea is the rather loose concept of "physically meaningful sentence." We will not try to define such a concept. However we presume (or at least hope) that our main examples somehow satisfy that criterion, as they deal with objects defined within physical theories, and consider problems formulated within the usual intuitively understood mathematical constructions of physics. Chaitin says that mathematics is random at its core. We show here that mathematics and the mathematically–based sciences are also pervaded by undecidability and by high–degree versions of incompleteness when axiomatized.

3.2. Hilbert's 6th Problem

When we discuss the possibility of giving physics an axiomatic treatment we delve into an old and important question about physical theories [12, 68]. The sixth problem in Hilbert's celebrated list of mathematical problems sketches its desirable contours [34]:

> *The Mathematical Treatment of the Axioms of Physics.*
>
> The investigations on the foundations of geometry suggest the problem: *to treat in the same manner, by means of axioms, those physical sciences in which mathematics plays an important part; in the first rank are the theory of probability and mechanics.*
>
> As to the axioms of the theory of probabilities, it seems to me to be desirable that their logical investigation be accompanied by a rigorous and satisfactory development of the method of mean values in mathematical physics, and in particular in the kinetic theory of gases.
>
> Important investigations by physicists on the foundations of mechanics are at hand; I refer to the writings of Mach. . . , Hertz. . . , Boltzmann. . . , and Volkman. . . It is therefore very desirable that the discussion of the foundations of mechanics be taken up by mathematicians also. Thus Boltzmann's work on the principles of mechanics suggests the problem of developing mathematically the limiting processes, those merely indicated, which lead from the atomistic view to the laws of continua. Conversely one might try to derive the laws of motion of rigid bodies by a limiting process from a system of axioms depending upon the idea of continuously varying conditions on a material filling all space continuously, these conditions being defined by parameters. For the question as to the equivalence of different systems of axioms is always of great theoretical interest.
>
> If geometry is to serve as a model for the treatment of physical axioms, we shall try first by a small number of axioms to include as large a class as possible of physical phenomena, and then by adjoining new axioms to arrive gradually at the more special theories. At the same time Lie's

> principle of subdivision can perhaps be derived from the profound theory of infinite transformation groups. The mathematician will have also to take account not only of those theories coming near to reality, but also, as in geometry, of all logically possible theories. We must be always alert to obtain a complete survey of all conclusions derivable from the system of axioms assumed.
>
> Further, the mathematician has the duty to test exactly in each instance whether the new axioms are compatible with the previous ones. The physicist, as his theories develop, often finds himself forced by the results of his experiments to make new hypotheses, while he depends, with respect to the compatibility of the new hypotheses with the old axioms, solely upon these experiments or upon a certain physical intuition, a practice which in the rigorously logical building up of a theory is not admissible. The desired proof of the compatibility of all assumptions seems to me also of importance, because the effort to obtain such a proof always forces us most effectively to an exact formulation of the axioms.

3.3. A review of axiomatization techniques

There are two widely known basic procedures to axiomatize a mathematically–based theory, such as theories in physics, economics or the ecology of competing species:

- We either use a syntactical approach à la Bourbaki [45]; or
- We follow a semantic approach, with the help of what we have called Suppes predicates.

Both methods are, in effect, essentially equivalent (see [8, 14, 62]). Mathematical structures and the Suppes set–theoretical predicates are of course essentially formulated in the language of set theory.

Our goal is to follow the programme sketched in Hilbert's 6th Problem. We propose here an axiomatic treatment for physics that encompasses the whole of classical physics — classical mechanics and classical field theories — plus first–quantized theories like Dirac's theory of the electron and its non–relativistic counterpart, Schrödinger's theory.

Hilbert stresses that we have to take into consideration:

> [...] not only [...] those theories coming near reality, but also, as in geometry, [...] all logically possible theories.

So we will delve here with incompleteness and other metamathematical phenomena in the realm of physics.[2]

Models

Notice that we deal here with different concepts for which we loosely employ the same word "model":

- As interpretation of a formal language, as in the domain which is usually called "theory of models." This is the usual meaning for model.
- As in "model of, or interpretation for a domain of the physical world." This is the customary informal meaning of the word for physicists, perhaps a synonym for "picture," or "description."

Context will make clear the sense we require for "model."

Physics: from informality to mathematical rigor

Our main goal here is to axiomatize portions of physics within the framework of axiomatic set theory. It is therefore interesting and worthwhile to follow the main historical development of formal treatments of physics in three central domains, namely electromagnetic theory, gravitation theory, and classical mechanics. We will point out in an informal vein how mathematical rigor evolved together with the conceptual development and clarification of those domains. The last step, that of a rigorous axiomatization, will be treated below.

The main point in our exposition is: physics, both classical and quantum, is here seen as an outcome, or as an extension of classical mechanics.[3] The Lagrangian and Hamiltonian formalisms, for systems of particles and then for fields, are seen as a basic, underlying construct that specializes to the several theories considered. A course in theoretical physics usually starts from an exposition of the Lagrangian and Hamiltonian (the so–called analytico–canonical) formalisms, show how they lead to a general formal

[2]Results that derive from the use of metamathematical techniques which are among other things applied to the mathematics that underlie physics had already been obtained by Maitland–Wright [49] in the early 1970s. That author investigated some aspects of the development of Solovay's mathematics [58], i. e., of a forcing model of set theory where a weakened version of the axiom of choice holds, as well as the axiom "every subset of the real line is Lebesgue–measurable." Among other interesting results, it is shown that the theory of Hilbert spaces based on that model does not coincide with the classical version. Explorations of forcing models in physics can be found in [6, 7, 46].

[3]This is the actual way most courses in theoretical physics are taught.

treatment of field theories, and then one applies those formalisms to electromagnetic theory, to Schrödinger's quantum mechanics — which is obtained out of geometrical optics and the eikonal equation, which in turn arise from Hamilton–Jacobi theory — and gravitation and gauge fields, which grow out of the techniques used in the formalism of electromagnetic theory. Here we use a variant of this approach.

Electromagnetism

The first conceptually unified view of electromagnetic theory is given in Maxwell's treatise, dated 1873 (for a facsimile of the 1891 edition see [50]).

Maxwell's treatment was given a more homogeneous, more compact notation by J. Willard Gibbs, and a sort of renewed presentation of Maxwell's main conceptual lines appears in the treatise by Sir James Jeans (1925, [38]). Next step is Stratton's textbook with its well–known list of difficult problems [59], and then Jackson's book, still the main textbook in the 1970s and 1980s [37].

When one looks at the way electromagnetic theory is presented in these books one sees that:

- The mathematical framework is calculus — the so–called advanced calculus, plus some knowledge of ordinary and partial differential equations — and linear algebra.
- Presentation of the theory's kernel becomes more and more compact; its climax is the use of covariant notation for the Maxwell equations. However covariant notation only appears as a development out of the set of Maxwell equations in the traditional Gibbsian "gradient–divergence–rotational" vector notation.

So, the main trend observed in the presentation of electromagnetic theory is: the field equations for electromagnetic theory are in each case summarized as a small set of coordinate–independent equations with a very synthetic notation system. When we need to do actual computations, we fall back into the framework of classical, 19th–century analysis, since for particular cases (actual, real–world, situations), the field equations open up in general to complicated, quite cumbersome differential equations to be solved by mostly traditional techniques.

A good reference for the early history of electromagnetism (even if its views of the subject matter are pretty heterodoxical) is O'Rahilly's tract

[52].

General relativity and gravitation

The field equations for gravitation we use today, that is, the Einstein field equations, are already born in a compact, coordinate–independent form (1915/1916) [30]. We find in Einstein's original presentation an explicit striving for a different kind of unification, that of a *conceptual* unification of all domains of physics. An unified formalism at that moment meant that one derived all different fields from a single, unified, fundamental field. That basic field then "naturally" splits up into the several component fields, very much like, or in the search of an analogy to, the situation uncovered by Maxwell in electromagnetism, where the electric field and the magnetic field are different concrete aspects of the same underlying unified electromagnetic field.

This trend starts with Weyl's theory [67] in 1918 just after Einstein's introduction in 1915 of his gravitation theory, and culminates in Einstein's beautiful, elegant, but physically unsound unified theory of the non–symmetric field (1946, see [29]). Weyl's ideas lead to developments that appear in the treatise by Corson (1953, [13]), and which arrive at the gauge field equations, or Yang–Mills equations (1954), which were for the first time examined in depth by Utiyama in 1956 [65].

An apparently different approach appears in the Kaluza–Klein unified theories. Originally unpromising and clumsy–looking, the blueprint for these theories goes back to Kaluza (1921) and then to Klein (1926, [63]). In its original form, the Kaluza–Klein theory is basically the same as Einstein's gravitation theory over a 5–dimensional manifold, with several artificial–looking constraints placed on the fifth dimension; that extra dimension is associated to the electromagnetic field.

The unpleasantness of having to deal with extraneous conditions that do not arise out of the theory itself was elegantly avoided when A. Trautmann in the late 1960s and then later Y. M. Cho, in 1975 [11], showed that the usual family of Kaluza–Klein–like theories arises out of a simile of Einstein's theory over a principal fiber bundle on spacetime with a semi–simple Lie group G as the fiber. Einstein's Lagrangian density over the principal fiber bundle endowed with its natural metric tensor splits up as Einstein's usual gravitational Lagrangian density with the so–called cosmological term plus an interacting gauge field Lagrangian density; depending on the group G

one gets electromagnetic theory, isospin theory, and so on. The cosmological constant arises in the Cho–Trautmann model out of the Lie group's structure constants, and thus gives a possible geometrical meaning to its interpretation as dark energy.

Here, conceptual unification and formal unification go hand in hand, but in order to do so we must add some higher–order objects (principal fiber bundles and the associated spaces, plus connections and connection forms) to get our more compact, unified treatment of gravitation together with gauge fields, which subsume the electromagnetic field. We are but a step away from a rigorous axiomatic treatment.

Classical mechanics

The first efforts towards an unification of mechanics are to be found in Lagrange's *Traité de Mécanique Analytique* (1811) and in Hamilton's results.

But one may see Hertz as the author of the first unified, mathematically well–developed presentation of classical mechanics in the late 1800s, in a nearly contemporary mathematical language. His last book, *The Principles of Mechanics*, published in 1894, advances many ideas that will later resurface not just in 20th century analytical mechanics, but also in general relativity [33]. Half a century later, in 1949, we have two major developments in the field: C. Lanczos publishes *The Variational Principles of Mechanics*, a brilliant mathematical essay [42] that for the first time presents classical mechanics from the unified viewpoint of differential geometry and Riemannian geometry. Concepts like kinetic energy or Coriolis force are made into geometrical constructs (respectively, Riemannian metric and affine connection); several formal parallels between mechanical formalism and that of general relativity are established. However the style of Lanczos' essay is still that of late 19th century and early 20th century mathematics, and is very much influenced by the traditional, tensor–oriented, over a local coordinate domain, presentations of general relativity.

New and (loosely speaking) higher–order mathematical constructs appear when Steenrod's results on fiber bundles and Ehresmann's concepts of connection and connection forms on principal fiber bundles are gradually applied to mechanics; those concepts go back to the late 1930s and early 1940s, and make their way into the mathematical formulations of mechanics in the late 1950s. Folklore has that the use of symplectic geometry in

mechanics first arose in 1960 when a major unnamed mathematician[4] circulated a letter among colleagues which formulated Hamiltonian mechanics as a theory of flows over symplectic manifolds, that is, a Hamiltonian flow is a flow that keeps invariant the symplectic form on a given symplectic manifold. The symplectic manifold was the old phase space; invariance of the symplectic form directly led to Hamilton's equations, to Liouville's theorem on the incompressibility of the phase fluid, and to the well–known Poincaré integrals — and here the advantage of a compact formalism was made clear, as the old, computational, very cumbersome proof for the Poincaré invariants was substituted for an elegant two–line, strictly geometrical proof.

High points in this direction are Sternberg's lectures (1964, [60]), MacLane's monograph (1968, [47]) and then the Abraham–Marsden treatise, *Foundations of Mechanics* [1]. Again one had at that moment a physical theory fully placed within the domain of a rigorous (albeit intuitive) mathematical framework, as in the case of electromagnetism, gauge field theory and general relativity. So, the path was open for an axiomatic treatment.

From classical to quantum mechanics

Quantum mechanics has always been snugly cradled in the classical theory, at least when considered by theoretical and mathematical physicists, far from the cloudy popular misconceptions that have surrounded the domain since its inception in the late 1920s. The Bohr–Sommerfeld quantization conditions in the first, "old," quantum theory, arise from the well–known Delaunay conditions in celestial mechanics; so much for the old quantum theory. The new, or Schrödinger–Heisenberg–Dirac quantum mechanics is nearly empirical in its inception [66], but when Schrödinger and Dirac appear on stage [23] we clearly see that the theory's conceptual roots and formalism arise out of classical mechanics. Schrödinger's wave equation is a kind of reinterpretation of the eikonal equation in geometrical optics, which in turn is a consequence of the Hamilton–Jacobi equation; the Dirac commutators and Heisenberg's motion equations are new avatars of well–known equations in the classical theory that involve Poisson brackets. We can also look at the motion equations:

$$\frac{dG}{dt} = \frac{\partial G}{\partial t} + \{H, G\}$$

[4]Said to be Richard Palais.

as the definition of a partial connection given by the Hamiltonian H on a manifold [40].

A surprising technical development stems from the efforts by Wightman to place quantum mechanics and the second–quantization theories on a firm mathematical ground. The starting point here was von Neumann's view in the early thirties that quantum mechanics was a linear dynamical theory of operators on some Hilbert space. The Murray and von Neumann theory of what we now know as von Neumann algebras (1936), later expanded to the theory of C^* algebras, allowed a group of researchers to frame several quantum–theoretic constructions in a purely algebraic way. Its realization in actual situations is given by a quantum state that induces a particular representation for the system (representation is here taken as the meaning used in group theory). This is the so–called Gelfand–Naimark–Segal construction [31].

The C^* algebra approach covers many aspects of quantum field theory, and is again framed within a rigorous, albeit intuitive mathematical background. It also exhibits some metamathematical phenomena, since the existence of some very general representations for C^* algebras are dependent of the full axiom of choice.

To sum it up: physics has strived for conceptual unification during the 20th century. This unification was attained in the domains we just described through a least–effort principle (Hamilton's Principle) applied to some kind of basic field, the Lagrangian or Lagrangian density, from which all known fields should be derived.

Most of physics is already placed on a firm mathematical ground, so that a strict axiomatic treatment of the main physical theories is possible. Still, there are mathematically uncertain procedures which are part of the everyday activity of the theoretical physicist, like Feynmann integration — but in this particular example we can take Feynmann's technique as an algorithm for the generation of a series of Feynmann diagrams, that is, as a strictly symbolic computational procedure. Other theoretical physics constructs that do not have a clear mathematical formulation (e.g. Boltzmann's H–theorem) can perhaps be approached in a similar way, as when we obtain formal series expansions out of the entropy integral, while one waits for a sound mathematical formulation for it.

3.4. Structures, species of structures, models

We introduce here the concept of a mathematical structure. Presentation will be mostly informal for the sake of clarity, and it is quite easy to develop the ideas introduced in a rigorous way. However we will go slightly beyond what is strictly required in this essay for the sake of completeness.

Mathematical structures are usually introduced either within the framework of set theory, or within higher–order logic, that is, type theory. Both presentations turn out to be essentially equivalent; professional mathematicians (see Bourbaki [8]) and model–theorists favor the first approach, while some logicians like Russell and Carnap explored the second way [9, 10, 55].

We follow here the set–theoretic approach. We will define mathematical structures within set theory, that is to say, mathematical structures will be conceived here as set–theoretic constructs.

Mathematical structures, species of mathematical structures

Very roughly: the structure is the arena where the game is to be played; the species of structures tells us the rules of the game.

To sum it up before we start and to give a first idea of the concepts we deal with here:

- A *mathematical structure* is finite sequence of sets, the *basic sets*, and some other sets which are obtained from the basic sets through a finite number of applications of two operations, the power set operation and the Cartesian product.
- A *species of mathematical structures* is a set–theoretic predicate which is the conjunction of two parts: first, a description of the structures we shall deal with, and second, the axioms that those structures must satisfy.

We suppose that we are working within some standard system of set theory, such as, say, Zermelo–Fraenkel set theory (ZF) [8, 32].

Suppes predicates

This presentation is semi–formal. We leave aside several important concepts like transportability [8, 14] for the sake of clarity in the exposition. The usual concept of species of mathematica structures as introduced by

Bourbaki [8] is a syntactical one. One of the authors formulated [14] a semantical version of it which is fully equivalent to the original construction. That second notion was called a *Suppes predicate* in the reference.

Loosely speaking, it can be described as follows. Let $\mathcal{L}_{\mathrm{ZF}}$ be the language of ZF set theory. We construct a predicate

$$P(S, X_0, X_1, \ldots, X_n),$$

that is to say, a formula of $\mathcal{L}_{\mathrm{ZF}}$ that defines a particular kind S of structures based on the sets $X_1, X_2, \ldots$. Predicate P is given by the conjunction of two pieces:

- First piece, $P_1(S, X_0, X_1, \ldots, X_n)$, shows how structure S is built out of the basic sets $X_0, X_1, \ldots, X_n$.
- Second piece, $P_2(S, X_0, X_1, \ldots, X_n)$, is the conjunction of the axioms that we wish S to satisfy.

We get:

$$P(S, X_0, X_1, \ldots, X_n) \leftrightarrow_{\mathrm{Def}} P_1(S, X_0, X_1, \ldots, X_n) \wedge P_2(S, X_0, X_1, \ldots, X_n).$$

Here $P(S, X_0, X_1, \ldots, X_n)$ is called *a species of structures on the basic sets*

$$X_0, \ldots, X_n,$$

and the predicate:

$$\exists X_0, X_1, \ldots, X_n \, P(S, X_0, X_1, \ldots, X_n)$$

is called the class of structures that corresponds to P.

3.5. Axiomatization in mathematics

The preceding construction sketches the required background for our formal treatment in this essay. It shows the way we will fit usual mathematical concepts like those of group or topological space within a formal framework like ZF set theory. We will in general assume that those formalizations have been done as in our examples; if required, each structure and species of structures we deal with can easily be made explicit (even if with some trouble). It is in general enough to know that we can axiomatize a theory of our interest with a Suppes predicate.

An axiomatic theory starts out of some primitive (undefined) concepts and out of a set of primitive propositions, the theory's axioms or postulates.

Other concepts are obtained by definition from the primitive concepts and from defined concepts; theorems of the theory are derived by proof mechanisms out of the axioms.

Given the set–theoretic viewpoint for our axiomatization, primitive concepts are sets which are related through the axioms. We adopt here the views expressed with the help of Suppes' slogan [62], slightly modified to suit our presentation:

> *To axiomatize a theory in mathematics, or in the mathematics–based sciences, is to define a set–theoretic predicate, that is to say, a species of structures.*

More precisely, to axiomatize an informal theory is to exhibit a species of structures so that:

- The primitive terms of the theory are the basic sets and the primitive relations of the species of structures.
- The theorems of the theory are the logical consequences of the species of structures, whose primitive elements are replaced by the corresponding primitive terms of the theory.
  Proofs are made within set theory.

Let P be the set–theoretic predicate that describes our theory. The structures which are models of P are a family $\mathcal{F}_P$ that may be taken to determine P. Therefore, in order to study the theory $\mathcal{T}_P$ given by P we can either proceed syntactically out of P, or semantically, out of $\mathcal{F}_P$.

Roughly, the syntactical and semantical approaches are complementary: given $\mathcal{F}_P$ we can recover P, and vice–versa. As we define P in set theory, and as set theory can be taken as a fully axiomatic theory [8, 39], then the theory of P–structures, which is the theory of $\mathcal{T}_P$ can also be formulated (and formalized) within set theory.

Therefore, any mathematical theory of the kind considered in physics as we have described it in this paper can in principle be formalized. That is to say: given any mathematical argument, such as the proof of any major theorem within mathematics, from Euclid's proofs to the prime–number distribution theorems, to the Malgrange preparation theorem or to the ergodic theorem, anything in mathematics can be formalized with the present techniques, either within ZF set theory or within one of its extensions.

What is an empirical theory?

We may identify theories in an obvious way with sets of sentences, to which we add an empirical counterpart, that is to say, observational terms, protocol sentences, and the like. Also, we may assert that a theory is a family of structures (models). Finally, according to our viewpoint a theory may also be identified to a triple $\langle \mathcal{S}, D, R \rangle$, where $\mathcal{S}$ is a species of structures (Suppes predicates), D the set of domains of applications, and R the rules that relate $\mathcal{S}$ to R [15, 16].

These views are not mutually incompatible.

We mainly adopt this third viewpoint. We will take here an empirical theory to be a species of structures plus a domain D and the set of rules R that relate the sentences to D. However we must distinguish the two steps required in the axiomatization of an empirical theory:

- Construction of the theory's Suppes predicate;
- Characterization of D and R, a procedure that depends on the science where we find that theory.

3.6. Suppes predicates for classical field theories in physics

The usual formal[5] treatment for physics, that is axiomatization in a general setting, goes as follows: one writes down a Lagrangian or a Lagrangian density for the phenomena we are interested at, and then use the variational principle as a kind of algorithmic procedure to derive the Euler–Lagrange equations, which give us the dynamics of the system. The variational principle also allows us to obtain a conservation–law, symmetry dependent, interpretation of interaction as in the case of the introduction of gauge fields out of symmetry conditions imposed on some field [13, 65].

We take here a slightly different approach. We describe the arena where physics happens — phase space, spacetime, fibered spaces — and add the dynamics through a Dirac–like equation.

Our results are not intended as a complete, all–encompassing, axiomatics for the whole of physics: there are many interesting areas in physics with uncertain mathematical procedures at the moment, such as statistical mechanics or quantum field theory, and the present techniques do not seem

[5]We will proceed in an informal way, and leave to the archetypical interested reader the toil and trouble of translating everything that we have done into a fully formal, rigorous, treatment of our presentation.

to be adequate for them. But we may confidently say that our axiomatization covers the whole of classical mechanics, classical field theory and first–quantized quantum mechanics.

We follow the usual mathematical notation in this subsection. In particular, Suppes predicates are written in a more familiar but essentially equivalent way.

The species of structures of essentially all classical physical theories can be formulated as particular dynamical systems derived from the triple $P = \langle X, G, \mu \rangle$, where X is a topological space, G is a topological group, and μ is a measure on a set of finite rank over $X \cup G$ and it is easy to put it in the form of a species of structures.

Thus we can say that the mathematical structures of physics arise out of the geometry of a topological space X. More precisely, physical objects are (roughly) the elements of X that:

- Exhibit invariance properties with respect to the action of G.
  (Actually the main species of structures in "classical" theories can be obtained out of two objects, a differentiable finite–dimensional real Hausdorff manifold M and a finite–dimensional Lie group G.)
- Are "generic" with respect to the measure μ for X.
  (This means, we deal with objects of probability 1. So, we only deal with "typical" objects, not the "exceptional" ones. This condition isn't always used, we must note, but anyway measure μ allows us to identify the exceptional situations in any construction.)

Let's now give all due details:

Definition 3.1. The species of structures of a **classical physical theory** is given by the 9–tuple

$$\Sigma = \langle M, G, P, \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G}, B, \nabla\varphi = \iota \rangle,$$

which is thus described:

(1) **The Ground Structures**. $\langle M, G \rangle$, where M is a finite–dimensional real differentiable manifold and G is a finite–dimensional Lie group.
(2) **The Intermediate Sets**. A fixed principal fiber bundle $P(M, G)$ over M with G as its fiber plus several associated tensor and exterior bundles.
(3) **The Derived Field Spaces**. Potential space $\mathcal{A}$, field space $\mathcal{F}$ and the current or source space $\mathcal{I}$. $\mathcal{A}$, $\mathcal{F}$ and $\mathcal{I}$ are spaces (in general,

manifolds) of cross–sections of the bundles that appear as intermediate sets in our construction.

(4) **Axiomatic Restrictions on the Fields**. The dynamical rule $\nabla\varphi = \iota$ and the relation $\varphi = d(\alpha)\alpha$ between a field $\varphi \in \mathcal{F}$ and its potential $\alpha \in \mathcal{A}$, together with the corresponding boundary conditions B. Here $d(\alpha)$ denotes a covariant exterior derivative with respect to the connection form α, and ∇ a covariant Dirac–like operator.

(5) **The Symmetry Group**. $\mathcal{G} \subseteq \mathrm{Diff}(M) \otimes \mathcal{G}'$, where $\mathrm{Diff}(M)$ is the group of diffeomorphisms of M and $\mathcal{G}'$ the group of gauge transformations of the principal bundle P.

(6) **The Space of Physically Distinguishable Fields**. If $\mathcal{K}$ is one of the $\mathcal{F}$, $\mathcal{A}$ or $\mathcal{I}$ field manifolds, then the space of physically distinct fields is $\mathcal{K}/\mathcal{G}$. □

(In more sophisticated analyses we must replace our concept of theory for a more refined one. Actually in the theory of science we proceed as in the practice of science itself by the means of better and better approximations. However for the goals of the present work our concept of empirical theory is enough.)

What we understand as the classical portion of physics up to the level of first–quantized theories easily fits into the previous scheme. We discuss in detail several examples: Maxwellian theory, Hamiltonian mechanics, general relativity and classical gauge field theory. Then, what is given an abstract form will receive its usual empirical translation.

Maxwell's electromagnetic theory

Let $M = \mathsf{R}^4$, with its standard differentiable structure. Let us endow M with the Cartesian coordination induced from its product structure, and let $\eta = \mathrm{diag}(-1, +1, +1, +1)$ be the symmetric constant metric Minkowskian tensor on M.

Then M is Minkowski spacetime, the physical arena where we do special relativity theory. As it is well–known, out of the linear transformations that keep invariant tensor η we obtain the well–known relativistic contraction and dilation phenomena.

We use standard physics notation. If the $F_{\mu\nu}(x)$ are components of the electromagnetic field, that is, a differentiable covariant 2–tensor field on M,

$\mu, \nu = 0, 1, 2, 3$, then Maxwell's equations are:

$$\partial_\mu F^{\mu\nu} = j^\nu,$$

$$\partial_\mu F_{\nu\rho} + \partial_\rho F_{\mu\nu} + \partial_\nu F_{\rho\mu} = 0.$$

The contravariant vectorfield whose components are given by the set of four smooth functions $j^\mu(x)$ on M is the current that serves as source for Maxwell's field $F_{\mu\nu}$. (We allow piecewise differentiable functions to account for shock–wave like solutions.)

It is known that Maxwell's equations are equivalent to the Dirac–like set

$$\nabla\varphi = \iota,$$

where

$$\varphi = (1/2)F_{\mu\nu}\gamma^{\mu\nu},$$

and

$$\iota = j_\mu\gamma^\mu,$$

$$\nabla = \gamma^\rho\partial_\rho,$$

(where the $\{\gamma^\mu : \mu = 0, 1, 2, 3\}$ are the Dirac gamma matrices with respect to η, that is, they satisfy the anticommutation rules $\gamma^\mu\gamma^\nu + \gamma^\nu\gamma\mu = 2\eta^{\mu\nu}$). Those equation systems are to be understood together with boundary conditions that specify a particular field tensor $F_{\mu\nu}$ "out of" the source j^ν [25].

The symmetry group of the Maxwell field equations is the Lorentz–Poincaré group that acts on Minkowski space M and in an induced way on objects defined over M. However since we are interested in *complex* solutions for the Maxwell system, we must find a reasonable way of introducing complex objects in our formulation. One may formalize the Maxwellian system as a gauge field. We sketch the usual formulation: again we start from $M = \langle \mathsf{R}^4, \eta\rangle$, and construct the trivial circle bundle $P = M \times S^1$ over M, since Maxwell's field is the gauge field of the circle group S^1 (usually written in that respect as $U(1)$). We form the set $\mathcal{E}$ of bundles associated to P whose fibers are finite–dimensional vectorspaces. The set of physical fields in our theory is obtained out of some of the bundles in $\mathcal{E}$: the set of electromagnetic field tensors is a set of cross–sections of the bundle $F = \Lambda^2 \otimes s^1(M)$ of all s^1–valued 2–forms on M, where s^1 is the group's Lie algebra. To be more precise, the set of all electromagnetic fields is

$\mathcal{F} \subset C^k(F)$, if we are dealing with C^k cross–sections (actually a submanifold in the usual C^k topology due to the closure condition $dF = 0$).

Finally we have two group actions on $\mathcal{F}$: the first one is the Lorentz–Poincaré action L which is part of the action of diffeomorphisms of M; then we have the (here trivial) action of the group $\mathcal{G}'$ of gauge transformations of P when acting on the field manifold $\mathcal{F}$. As it is well known, its action is *not* trivial in the non–Abelian case. Anyway it always has a nontrivial action on the space $\mathcal{A}$ of all gauge potentials for the fields in $\mathcal{F}$. Therefore we take as our symmetry group $\mathcal{G}$ the product $L \otimes \mathcal{G}'$ of the (allowed) symmetries of M and the symmetries of the principal bundle P.

We must also add the spaces $\mathcal{A}$ of potentials and of currents, $\mathcal{I}$, as structures derived from M and S^1. Both spaces have the same underlying topological structure; they differ in the way the group $\mathcal{G}'$ of gauge transformations acts upon them. We obtain $I = \Lambda^1 \otimes s^1(M)$ and $\mathcal{A} = \mathcal{I} = C^k(I)$. Notice that $\mathcal{I}/\mathcal{G}' = \mathcal{I}$ while $\mathcal{A}/\mathcal{G}' \neq \mathcal{A}$.

Therefore we can say that the 9–tuple

$$\langle M, S^1, P, \mathcal{F}, \mathcal{A}, \mathcal{G}, \mathcal{I}, B, \nabla\varphi = \iota \rangle$$

where M is Minkowski space, and B is a set of boundary conditions for our field equations $\nabla\varphi = \iota$, represents the species of mathematical structures of a Maxwellian electromagnetic field, where P, $\mathcal{F}$ and $\mathcal{G}$ are derived from M and S^1. The Dirac–like equation

$$\nabla\varphi = \iota$$

should be seen as an axiomatic restriction on our objects; the boundary conditions B are (i) a set of derived species of structures from M and S^1, since, as we are dealing with Cauchy conditions, we must specify a local or global spacelike hipersurface C in M to which (ii) we add sentences of the form $\forall x \in C\, f(x) = f_0(x)$, where f_0 is a set of (fixed) functions and the f are adequate restrictions of the field functions and equations to C.

Consistency of the added axioms

Hamiltonian mechanics

Hamiltonian mechanics is here seen as the dynamics of the "Hamiltonian fluid" [1, 3, 42]. Our ground structure for mechanics starts out of basic sets which are a $2n$–dimensional real smooth manifold, and the real symplectic group $\mathrm{Sp}(2n, \mathsf{R})$. Phase spaces in Hamiltonian mechanics are symplectic

manifolds: even–dimensional manifolds like M endowed with a symplectic form, that is, a nondegenerate closed 2–form Ω on M. The imposition of that form can be seen as the choice of a reduction of the linear bundle $L(M)$ to a fixed principal bundle $P(M, \mathrm{Sp}(2n, \mathsf{R}))$; however given one such reduction it doesn't automatically follow that the induced 2–form on M is a closed form.

All other objects are constructed in about the same way as in the preceding example. However we must show that we still have here a Dirac–like equation as the dynamical axiom for the species of structures of mechanics. Hamilton's equations are

$$i_X \Omega = -dh,$$

where i_X denotes the interior product with respect to the vectorfield X over M, and h is the Hamiltonian function. That equation is (locally, at least) equivalent to:

$$L_X \Omega = 0,$$

or

$$d(i_X \Omega) = 0,$$

where L_X is the Lie derivative with respect to X. The condition $d\varphi = 0$, with $\varphi = i_X \Omega$, is the degenerate Dirac–like equation for Hamiltonian mechanics. We don't get a full Dirac–like operator $\nabla \neq d$ because M, seen as a symplectic manifold, doesn't have a canonical metrical structure, so that we cannot define (through the Hodge dual) a canonical divergence δ dual to d. The group that acts on M with its symplectic form is the group of canonical transformations; it is a subgroup of the group of diffeomorphisms of M so that symplectic forms are mapped onto symplectic forms under a canonical transformation. We can take as "potential space" the space of all Hamiltonians on M (which is a rather simple function space), and as "field space" the space of all "Hamiltonian fields" of the form $i_X \Omega$.

Interpretations are immediate here: h is the system's Hamiltonian, which (given some simple conditions) can be seen as the system's total energy. Invariance of the symplectic form by the Lie derivative with respect to a Hamiltonian flow is equivalent both to Poincaré's integral invariant theorem and to Liouville's theorem — just as a flavor of the way our treatment handles well–known concepts and results in mechanics.

General relativity

General relativity is a theory of gravitation that interpretes this basic force as originated in the pseudo–Riemannian structure of spacetime. That is to say: in general relativity we start from a spacetime manifold (a 4–dimensional, real, adequately smooth manifold) which is endowed with an pseudo–Riemannian metric tensor. Gravitational effects originate in that tensor.

Given any 4–dimensional, noncompact, real, differentiable manifold M, we can endow it with an infinite set of different, nonequivalent pseudo–Riemannian metric tensors with a Lorentzian signature (that is, $-+++$). That set is uncountable and has the power of the continuum. (By nonequivalent metric tensors we mean the following: form the set of all such metric tensors and factor it by the group of diffeomorphisms of M; we get a set that has the cardinality of the continuum. Each element of the quotient set is a different gravitational field for M.)

Therefore, neither the underlying structure of M as a topological manifold, nor its differentiable structure determines a particular pseudo–Riemannian metric tensor, that is, a specific gravitational field. From the strictly geometrical viewpoint, when we choose a particular metric tensor g of Lorentzian signature, we determine a g–dependent reduction of the general linear tensor bundle over M to one of its pseudo–orthogonal bundles. The relation

$$g \mapsto g\text{–dependent reduction of the linear bundle to a pseudo–orthogonal bundle}$$

is 1–1.

We now follow our recipe:

- We take as basic sets a 4–dimensional real differentiable manifold of class C^k, $1 \leq k \leq +\infty$, and the Lorentz pseudo–orthogonal group $O(3,1)$.
- We form the principal linear bundle $L(M)$ over M; that structure is solely derived from M, as it arises from the covariance properties of the tangent bundle over M. From $L(M)$ we fix a reduction of the bundle group $L(M) \rightarrow P(M, O(3,1))$, where $P(M, O(3,1))$ is the principal fiber bundle over M with the $O(3,1)$ group as its fiber.
  Those will be our derived sets. We therefore inductively define a Lorentzian metric tensor g on M, and get the couple $\langle M, g \rangle$, which is

spacetime.
(Notice that the general relativity spacetime arises quite naturally out of the interplay between the theory's "general covariance" aspects, which appear in $L(M)$, and — as we will see in the next section — its "gauge–theoretic features, which are clear in $P(M, O(3,1))$.)

- Field spaces are:
  - The first is the set (actually a manifold, with a natural differentiable structure) of all pseudo–Riemannian metric tensors, $\mathcal{M} \subset C^k(\odot^2 T_*(M))$, where $C^k(\odot^2 T_*(M))$ is the bundle of all C^k symmetric covariant 2–tensors over M.
  - Also out of M and out of adequate associated bundles we get $\mathcal{A}$, the bundle of all Christoffel connections over M, and $\mathcal{F}$, the bundle of all Riemann–Christoffel curvature tensors over M.
- We need the space of source fields, $\mathcal{I}$, that includes energy–momentum tensors, and arise out of adequate associated tensor bundles over M.
- $\mathcal{G}$ is the group of C^k–diffeomorphisms of M.
- If $\mathcal{K}$ is any of the field spaces above, then $\mathcal{K}/\mathcal{G}$ is the space of physically distinct fields.
- Finally the dynamics are given by Einstein's equations (there is also a Dirac–like formulation for those, first proposed by R. Penrose in 1960 as a neutrino–like equation; see [24]).

The quotient $\mathcal{K}/\mathcal{G}$ is the way we distinguish concrete, physically diverse, fields, as for covariant theories one has that any two fields related by an element of $\mathcal{G}$ "are" the "same" field.

Classical gauge fields

The mathematics of classical gauge fields can be found in [5, 65]. We follow here the preceding examples, and in particular the treatment of general relativity:

- The basic sets are a spacetime $\langle M, g\rangle$, and a finite dimensional, semi–simple, compact Lie group G.
- The derived set is a fixed principal bundle $P(M, G)$ over M with G as the fiber.
- The group of gauge transformations $\mathcal{G}$ is the subgroup of all diffeomorphisms of $P(M, G)$ that reduce to a diffeomorphism on M and to the group action on the fiber.
- If $\ell(G)$ is the Lie algebra of G, we get:

  - Connection–form space, or the space of potentials, noted $\mathcal{A}$, is the space of all C^k–cross sections of the bundle of $\ell(G)$–valued 1–forms on M.
  - Curvature space, or the space of fields $\mathcal{F}$, is the space of all C^k–cross sections of $\ell(G)$–valued 2–forms on M, such that $F \in \mathcal{F}$ is the field with potential $A \in \mathcal{A}$.
  - Source space $\mathcal{I}$ coincides with $\mathcal{A}$, but is acted upon in a different way by the group $\mathcal{G}$ of gauge transformations. (Currents in $\mathcal{I}$ are *tensorial* 1–forms, while gauge–potentials in $\mathcal{A}$ are transformed via an inhomogeneous transformation.)

- The space of physically different fields is $\mathcal{K}/\mathcal{G}$, where $\mathcal{K}$ is any of the above field spaces.
- Dynamics are given by the usual gauge–field equations, which are a nonlinear version of the electromagnetic field equations. There is also a Dirac–like equation for gauge fields [27].

To sum it up: with the help of the schema presented at the beginning of the section, we can say that the structure of a physical theory is an ordered pair $\langle \mathcal{F}, \mathcal{G} \rangle$, where $\mathcal{F}$ is an infinite–dimensional space of fields, and $\mathcal{G}$ is an infinite–dimensional group that acts upon field space. To get the Suppes predicate we must add the information about the dynamical equations $D(\phi) = 0, \phi \in \mathcal{F}$, for the fields ϕ.

Notice that general relativity can be seen as a kind of degenerate gauge field theory, more precisely a gauge theory of the $O(3,1)$ group.

Quantum theory of the electron

The Dirac electron theory (and the general theory of particles with any spin) can be easily formalized according to the preceding schemata. One uses as geometrical background the setting for special relativity; dynamics is given either by Dirac's equation or Weyl's equation, for the case of zero–mass particles. Higher spin fields are dealt with the help either of the Bargmann–Wigner equations or their algebraic counterpart [25]. The Schrödinger equation is obtained from the Dirac set out of a — loosely speaking — 'standard' limiting procedure, which can be formally represented by the addition of new axioms to the corresponding Suppes predicate.

General field theory

Sometimes one may wish to discuss field theory in a very general, motion–equation independent, way. We then use as geometrical background the construction of Minkowski space and take as dynamical axioms the field–theoretic Euler–Lagrange equations, or, as we've said, we can take the variational principle as a formal algorithm to derive the dynamics of the system.

Other domains of science

We can extend the preceding techniques to several scientific domains. For example, the bulk of economics, as presented in Samuelson's *Foundations of Economic Analysis* [56], or some specific results, such as the Nash equilibrium theorem [18], easily fit within our construction — we can find in a straightforward way a Suppes predicate for results in mathematical economics [17]. The same goes with mathematical biology [44].

Summing it up

We have proceeded from start with a specific goal in mind: we wished to follow Hilbert's programme in his 6th Problem, that is, we proposed an axiomatiation of physics that allows us to explore many interesting mathematical consequences of those theories.

We now wish to obtain examples of Gödel sentences — undecidable sentences — within the axiomatic versions of those theories, and in a more general cadre, we wish to see the effects and consequences of metamathematical results and techniques when applied to those theories, or to their axiomatic versions.

3.7. Generalized incompleteness

Preliminary

For concepts from logic see [51]. We use: $\neg$, "not," $\vee$, "or," $\wedge$, "and," $\rightarrow$, "if... then...," $\leftrightarrow$, "if and only if," $\exists x$, "there is a x," $\forall x$, "for every x." $P(x)$ is a formula with x free; it roughly means "x has property P." Finally $T \vdash \xi$ means T proves ξ, or ξ is a theorem of T. ω is the set of natural numbers, $\omega = \{0, 1, 2, \ldots\}$.

We deal here mainly with algorithmic functions. These are given by their programs coded in Gödel numbers e [54]. We will sometimes use Turing machines (noted by sans–serif letters with the Gödel number as index M_e) or partial recursive functions, noted $\{e\}$.

We start from arithmetic, specifically from Peano Arithmetic, noted PA [51]. Its language includes variables $x, y, \ldots$, two constants, **0** and **1**, the equality sign $=$, and two operation signs, $+, \times$. We will also require Russell's ι symbol [41]. $\iota_x\, P(x)$ is, roughly, the x such that $P(x)$.

The *standard interpretation* for PA is: the variables $x, y, \ldots$ range over the natural numbers, and **0** and **1** are seen as, respectively, zero and one. PA is strong enough to formally include most of Turing machine theory [18, 19]. Recall that a Turing machine is given by its Gödel number, which recursively codes the machine's program. Rigorously, for PA, we have:

Definition 3.2. A Turing machine of Gödel number e operating on x with output y, $\{e\}(x) = y$ is **representable** in PA if there is a formula $F_e(x, y)$ in the language of our arithmetic theory so that:

(1) $\text{PA} \vdash F_e(x, y) \wedge F_e(x, z) \rightarrow y = z$, and
(2) For natural numbers a, b, $\{e\}(a) = b$ if and only if $\text{PA} \vdash F_e(a, b)$. □

Then we have the representation theorem for partial recursive functions in PA:

Prop 3.3. Every Turing machine is representable in Peano Arithmetic. Moreover there is an effective procedure that allows us to obtain F_e from the Gödel number e. □

Remark 3.4. We mainly consider here theories that are *arithmetically sound*, that is, which have a model with standard arithmetic for its arithmetical segment. □

A first example of generalized incompleteness

The example we now give shows that incompleteness is a pervasive phenomenon, from an arithmetic theory like PA and upwards, that is, it affects all theories that contain enough arithmetic, have a model where arithmetic is standard, and have a recursively enumerable set of theorems.

Suppose that our theory S has Russell's description symbol ι [41]. Let P be a predicate symbol so that for closed terms $\xi, zeta$ so that $S \vdash \xi \neq \zeta$,

$S \vdash P(\xi)$ and $S \vdash \neg P(\zeta)$ (we call such P, nontrivial predicates). Then, for the term:

$$\eta = \iota_x[(x = \xi \wedge \alpha) \vee (x = \zeta \wedge \neg\alpha)],$$

where α is an undecidable sentence in S:

Prop 3.5. $S \nvdash P(\eta)$ and $S \nvdash \neg P(\eta)$. □

This shows that incompleteness is found everywhere within theories like S.

Remark 3.6. From now on we will consider theories S, T, like the one characterized above. □

Our main tool here will be an explicit expression for the Halting Function, that is, the function that settles the halting problem [54]. We have shown elsewhere that it can be constructed within the language of classical analysis. We have originally used the Richardson transforms (see details and references in [18]) in order to obtain an explicit expression for the Halting Function, but they are not essential in our construction. We start from a strengthening of Proposition 3.3:

Prop 3.7. If $\{e\}(a) = b$, for natural numbers a, b, then we can algorithmically construct a polynomial p_e over the natural numbers so that $\{e\}(a) = b \leftrightarrow \exists x_1, x_2, \ldots, x_k \in \omega\, p_e(a, b, x_1, x_2, \ldots, x_k) = 0$. □

Prop 3.8. $a \in R_e$, where R_e is a recursively enumerable set, if and only if there are e and p so that $\exists x_1, x_2, \ldots, x_k \in \omega\, (p_e(a, x_1, x_2, \ldots, x_k) = 0)$. □

Our results derive from the preceding propositions.

The Halting Function

One of the main results in Alan Turing's great 1937 paper, "On computable numbers, with an application to the Entscheidungsproblem" [64], is a proof of the algorithmic unsolvability of a version of the halting problem: given an arbitrary Turing machine of Gödel number e, for input x, there is no algorithm that decides whether $\{e\}(x)$ stops and outputs something, or enters an infinite loop.

Remark 3.9. Let $\mathsf{M}_m(a) \downarrow$ mean: "Turing machine of Gödel number m stops over input a and gives some output." Similarly $\mathsf{M}_m(a) \uparrow$ means,

"Turing machine of Gödel number m enters an infinite loop over input a." Then we can define the halting function θ:

- $\theta(m, a) = 1$ if and only if $\mathsf{M}_m(a) \downarrow$.
- $\theta(m, a) = 0$ if and only if $\mathsf{M}_m(a) \uparrow$.

$\theta(m, a)$ is the halting function for M_m over input a. □

θ isn't algorithmic, of course [54, 64], that is, there is no Turing machine that computes it.

Then, if σ is the sign function, $\sigma(\pm x) = \pm 1$ and $\sigma(0) = 0$:

Expressions for the Halting Function

Prop 3.10 (The Halting Function.). *The halting function $\theta(n, q)$ is explicitly given by:*

$$\theta(n, q) = \sigma(G_{n,q}),$$

$$G_{n,q} = \int_{-\infty}^{+\infty} C_{n,q}(x) e^{-x^2} dx,$$

$$C_{m,q}(x) = |F_{m,q}(x) - 1| - (F_{m,q}(x) - 1).$$

$$F_{n,q}(x) = \kappa_P p_{n,q}. \square$$

Here $p_{n,q}$ is the two–parameter universal Diophantine polynomial and κ_P an adequate Richardson transform.

The succession of definite integrals

$$K(m) = \int_{-\infty}^{+\infty} \frac{C(m, x) e^{-x^2}}{1 + C(m, x)} dx,$$

also gives us the Halting Function:

$$\theta(m, x) = \theta(\langle m, x \rangle) = \theta(m) = \sigma(\frac{K_m}{1 + K_m}). \square$$

Remark 3.11. There is an expression for the Halting Function even within a simple extension of PA. Let $p(n, \mathbf{x})$ be a 1–parameter universal polynomial; $\mathbf{x}$ abbreviates $x_1, \ldots, x_p$. Then either $p^2(n, \mathbf{x}) \geq 1$, for all $\mathbf{x} \in \omega^p$, or there are $\mathbf{x}$ in ω^p such that $p^2(n, \mathbf{x}) = 0$ sometimes. As $\sigma(x)$ when restricted to ω is primitive recursive, we may define a function $\psi(n, \mathbf{x}) = 1 - \sigma p^2(n, \mathbf{x})$ such that:

- Either for all $\mathbf{x} \in \omega^p$, $\psi(n, \mathbf{x}) = 0$;
- Or there are $\mathbf{x} \in \omega^p$ so that $\psi(n, \mathbf{x}) = 1$ sometimes.

Thus the Halting Function can be represented as:

$$\theta(n) = \sigma[\sum_{\tau^q(\mathbf{x})} \frac{\psi(n, \mathbf{x})}{\tau^q(\mathbf{x})!}],$$

where $\tau^q(\mathbf{x})$ denotes the positive integer given out of $\mathbf{x}$ by the pairing function τ: if τ^q maps q–tuples of positive integers onto single positive integers, $\tau^{q+1} = \tau(x, \tau^q(\mathbf{x}))$. □

Undecidability and incompleteness

Our main undecidability (and the related incompleteness) results stem from the following:

Lemma 3.12. *There is a Diophantine set D so that*

$$m \in D \leftrightarrow \exists x_1, \ldots, x_n \in \omega \, p(m, x_1, \ldots, x_n) = 0,$$

p a Diophantine polynomial, and D is recursively enumerable but not recursive. □

Corollary 3.13. *For an arbitrary $m \in \omega$ there is no general decision procedure to check whether $p(m, x_1, \ldots) = 0$ has a solution in the positive integers.* □

Main undecidability and incompleteness result

Therefore, given such a p, and $F = \kappa_P(p)$, where κ_P is an adequate Richardson transform:

Corollary 3.14. *For an arbitrary $m \in \omega$ there is no general decision procedure to check whether, for F and G adequate real–defined and real–valued functions:*

(1) There are real numbers $x_1, \ldots, x_n$ such that $F(m, x_1, \ldots, x_n) = 0$;
(2) There is a real number x so that $G(m, x) < 1$;
(3) Whether we have $\forall x \in \mathsf{R} \; \theta(m, x) = 0$ or $\forall x \in \mathsf{R} \; \theta(m, x) = 1$ over the reals.
(4) Whether for an arbitrary $f(m, x)$ we have $f(m, x) \equiv \theta(m, x)$.

Proof: From the preceding results. The last undecidability statement follows from the third one. □

We conclude with a first, quite all–encompassing, result. Let $\mathcal{B}$ be a sufficiently large algebra of functions and let $P(x)$ be a nontrivial predicate. If ξ is any word in that language, we write $\|\xi\|$ for its complexity, as measured by the number of letters from ZFC's alphabet in ξ. Also we define the *complexity of a proof* $C_{\mathrm{ZFC}}(\xi)$ of ξ in the language of ZFC to be the minimum length that a deduction of ξ from the ZFC axioms can have, as measured by the total number of letters in the expressions that belong to the proof.

Remark 3.15. Recall that theory $T \supset$ PA is *arithmetically sound* if T has a model where its arithmetic is standard. □

Prop 3.16. If ZFC is arithmetically sound, then:

(1) There is an $h \in \mathcal{B}$ so that neither ZFC $\nvdash \neg P(h)$ nor ZFC $\nvdash P(h)$, but $\mathbf{N} \models P(h)$, where $\mathbf{N}$ makes ZFC arithmetically sound.
(2) There is a denumerable set of functions $h_m(x) \in \mathcal{B}$, $m \in \omega$, such that there is no general decision procedure to ascertain, for an arbitrary m, whether $P(h_m)$ or $\neg P(h_m)$ is provable in ZFC.
(3) Given the set $K = \{m : \mathrm{ZFC} \vdash \phi(\widehat{m})\}$, and given an arbitrary total recursive function $g : \omega \to \omega$, there is an infinite number of values for m so that $C_{\mathrm{ZFC}}(P(\widehat{m})) > g(\|P(\widehat{m})\|)$.

Proof: Let θ be as above. Let f_0, g_0 satisfy our conditions on P, that is, ZFC $\vdash P(f_0)$ and ZFC $\vdash \neg P(g_0)$. Then define:

$$h(m, x) = \theta(m, x) f_0 + (1 - \theta(m, x)) g_0.$$

This settles (2). Now let us specify θ so that the corresponding Diophantine equation $p = 0$ is never solvable in the standard model for arithmetic, while that fact cannot be proved in ZFC. We then form, for such an indicator function,

$$h = \theta f_0 + (1 - \theta) g_0.$$

This settles (1). Finally, for (3), we notice that as K is recursively enumerable but not recursive, it satisfies the conditions in the Gödel–Ehrenfeucht–Mycielski theorem about the length of proofs. □

3.8. Higher degrees

Our main result in this section is:

Prop 3.17. If T is arithmetically sound then we can explicitly and algorithmically construct in the language $\mathcal{L}_T$ of T an expression for the characteristic function of a subset of ω of degree $\mathbf{0''}$.

Remark 3.18. That expression depends on recursive functions defined on ω and on elementary real–defined and real–valued functions plus the absolute value function, a quotient and an integration, or perhaps an infinite sum, as in the case of the β and θ functions associated to the halting problem. □

Proof: We could simply use Theorem 9-II in [54] (p. 132). However for the sake of clarity we give a detailed albeit informal proof. Actually the degree of the set described by the characteristic function whose expression we are going to obtain will depend on the fixed oracle set A; so, our construction is a more general one.

Let us now review a few concepts. Let $A \subset \omega$ be a fixed infinite subset of the integers:

Definition 3.19. The **jump of** A is noted A'; $A' = \{x : \phi_x^A(x) \downarrow\}$, where ϕ_x^A is the A–partial recursive algorithm of index x. □

In order to make things self–contained, we review here some ideas about A–partial recursive functions.

From Turing machines to oracle Turing machines

(1) An oracle Turing machine ϕ_x^A with oracle A can be visualized as a two–tape machine where tape 1 is the usual computation tape, while tape 2 contains a listing of A. When the machine enters the oracle state s_0, it searches tape 2 for an answer to a question of the form "does $w \in A$?" Only finitely many such questions are asked during a converging computation; we can separate the positive and negative answers into two disjoint finite sets $D_u(A)$ and $D_v^*(A)$ with (respectively) the positive and negative answers for those questions; notice that $D_u \subset A$, while $D_v^* \subset \omega - A$. We can view those sets as ordered k– and k^*–ples; u and v are recursive codings for them [54]. The $D_u(A)$ and $D_v^*(A)$ sets can be coded as follows: only finitely many elements of A are queried during

an actual converging computation with input y; if k' is the highest integer queried during one such computation, and if $d_A \subset c_A$ is an initial segment of the characteristic function c_A, we take as a standby for D and D^* the initial segment d_A where the length $l(d_A) = k' + 1$.
We can effectively list all oracle machines with respect to a fixed A, so that, given a particular machine we can compute its index (or Gödel number) x, and given x we can recover the corresponding machine.

(2) Given an A–partial recursive function ϕ_x^A, we form the oracle Turing machine that computes it. We then do the computation $\phi_x^A(y) = z$ that outputs z. The initial segment $d_{y,A}$ is obtained during the computation.

(3) The oracle machine is equivalent to an ordinary two–tape Turing machine that takes as input $\langle y, d_{y,A}\rangle$; y is written on tape 1 while $d_{y,A}$ is written on tape 2. When this new machine enters state s_0 it proceeds as the oracle machine. (For an ordinary computation, no converging computation enters s_0, and $d_{y,A}$ is empty.)

(4) The two–tape Turing machine can be made equivalent to a one–tape machine, where some adequate coding places on the single tape all the information about $\langle y, d_{y,A}\rangle$. When this third machine enters s_0 it scans $d_{y,A}$.

(5) We can finally use the standard map τ that codes n–ples 1–1 onto ω and add to the preceding machine a Turing machine that decodes the single natural number $\tau(\langle y, d_{y,A}\rangle)$ into its components before proceeding to the computation.

Let w be the index for that last machine; we note the machine ϕ_w.

If x is the index for ϕ_x^A, we write $w = \rho(x)$, where ρ is the effective 1–1 procedure above described that maps indices for oracle machines into indices for Turing machines. Therefore,

$$\phi_x^A(y) = \phi_{\rho(x)}(\langle y, d_{y,A}\rangle).$$

Now let us note the universal polynomial $p(n, q, x_1, \ldots, x_n)$. We can define the jump of A as follows:

$$A' = \{\rho(z) : \exists x_1, \ldots, x_n \in \omega\, p(\rho(z), \langle z, d_{z,A}\rangle, x_1, \ldots, x_n) = 0\}.$$

With the help of the Richardson map described above, we can now form a function modelled after the θ function that settles the Halting Problem; it is the desired characteristic function:

$$c_{\emptyset'}(x) = \theta(\rho(x), \langle x, d_{x,\emptyset'}\rangle).$$

(Actually we have proved more; we have obtained

$$c_{A'}(x) = \theta(\rho(x), \langle x, d_{x,A}\rangle),$$

with reference to an arbitrary $A \subset \omega$.)

Finally, we write $\theta^{(2)}(x) = c_{\emptyset''}(x)$. □

We recall [54]:

Definition 3.20. The complete Turing degrees $\mathbf{0}, \mathbf{0}', \mathbf{0}'', \ldots, \mathbf{0}^{(p)}, \ldots$, $p < \omega$, are Turing equivalence classes generated by the sets $\emptyset, \emptyset', \emptyset'', \ldots, \emptyset^{(p)}, \ldots$. □

Now let $\mathbf{0}^{(n)}$ be the n–th complete Turing degree in the arithmetical hierarchy. Let $\tau(n, q) = m$ be the pairing function in recursive function theory [54]. For $\theta(m) = \theta(\tau(n, q))$, we have:

Corollary 3.21 (Complete Degrees.). *If T is arithmetically sound, for all $p \in \omega$ the expressions $\theta^p(m)$ explicitly constructed below represent characteristic functions in the complete degrees $\mathbf{0}^{(p)}$.*

Proof: From Proposition 3.17,

$$\begin{cases} \theta^{(0)} = c_{\emptyset}(m) = 0, \\ \theta^{(1)}(m) = c_{\emptyset'}(m) = \theta(m), \\ \theta^{(n)}(m) = c_{\emptyset^{(n)}}(m), \end{cases}$$

for c_A as in Proposition 3.17. □

Incompleteness theorems

We now state and prove several incompleteness results about axiomatized versions of arithmetic with a classical first–order language, a recursive vocabulary and a recursively enumerable set of axioms; say, Peano Arithmetic (PA). These results are of course also valid for extensions of it T with the same language and recursively enumerable set of axioms.

We suppose, as already stated, that PA $\subset T$ means that there is an interpretation of PA in T.

The next results will be needed when we consider our main examples. We recall that "$\dot{-}$" — the truncated sum — is a primitive recursive operation on ω:

- For $a > b$, $a \stackrel{\bullet}{-} b = a - b$.
- For $a < b$, $a \stackrel{\bullet}{-} b = 0$.

In the next result, Z is the set of integers. The starting point is the following consequence of a well–known result which we now quote: let $\mathbf{N}$ be a model, $\mathbf{N} \models T$, and $\mathbf{N}$ makes T arithmetically sound. Then:

Prop 3.22. If T is arithmetically sound, then we can algorithmically construct a polynomial expression $q(x_1, \ldots, x_n)$ over Z such that $\mathbf{M} \models \forall x_1, \ldots, x_n \in \omega\, q(x_1, \ldots, x_n) > 0\}$, but

$$T \not\vdash \forall x_1, \ldots, x_n \in \omega\, q(x_1, \ldots, x_n) > 0$$

and

$$T \not\vdash \exists x_1, \ldots, x_n \in \omega\, q(x_1, \ldots, x_n) = 0.$$

Proof: Let $\xi \in \mathcal{L}_T$ be an undecidable sentence obtained for T with the help of Gödel's diagonalization; let n_ξ be its Gödel number and let m_T be the Gödel coding of proof techniques in T (of the Turing machine that enumerates all the theorems of T). For an universal polynomial $p(m, q, x_1, \ldots, x_n)$ we have:

$$q(x_1, \ldots, x_n) = (p(m_T, n_\xi, x_1, \ldots, x_n))^2. \square$$

Corollary 3.23. *If* PA *is consistent then we can find within it a polynomial p as in Proposition 3.22.* $\square$

We can also state and prove a weaker version of Proposition 3.22:

Prop 3.24. If T is arithmetically sound, there is a polynomial expression over Z $p(x_1, \ldots, x_n)$ such that $\mathbf{N} \models \forall x_1, \ldots, x_n \in \omega\, p(x_1, \ldots, x_n) > 0$, while

$$T \not\vdash \forall x_1, \ldots, x_n \in \omega\, p(x_1, \ldots, x_n) > 0$$

and

$$T \not\vdash \exists x_1, \ldots, x_n \in \omega\, p(x_1, \ldots, x_n) = 0.$$

Proof: See [21]. If $p(m, x_1, \ldots, x_n)$, $m = \tau\langle q, r\rangle$, is an universal polynomial with τ being Cantor's pairing function [54], then $\{m : \exists x_1 \ldots \in \omega\, p(m, x_1, \ldots) = 0\}$ is recursively enumerable but not recursive. Therefore there must be an m_0 such that $\forall x_1 \ldots \in \omega\, (p(m_0, x_1, \ldots))^2 > 0$. (This

is actually a version of Post's original argument for the proof of Gödel's theorem [53].) □

Prop 3.25. If PA is consistent and $\mathbf{N} \models$ PA is standard, and if P is nontrivial then there is a term–expression $\zeta \in \mathcal{L}_{\rm PA}$ such that $\mathbf{N} \models P(\zeta)$ while PA $\nvdash P(\zeta)$ and PA $\nvdash \neg P(\zeta)$.

Proof: Put $\zeta = \xi + r(x_1, \ldots, x_n)\nu$, for $r = 1 \overset{\bullet}{-} (q+1)$, q as in Proposition 3.22 (or as p in Proposition 3.24). □

Remark 3.26. Therefore every nontrivial arithmetical P in theories from formalized arithmetic upwards turns out to be undecidable. We can generalize that result to encompass other theories T that include arithmetic; see below. □

3.9. The θ function and the arithmetical hierarchy

We now give alternative proofs for well–known results about the arithmetical hierarchy that will lead to other incompleteness results. Recall;

Definition 3.27. The sentences $\xi, \zeta \in \mathcal{L}_T$ are **demonstrably equivalent** if and only if $T \vdash \xi \leftrightarrow \zeta$. □

Definition 3.28. The sentence $\xi \in \mathcal{L}_T$ is **arithmetically expressible** if and only if there is an arithmetic sentence ζ such that $T \vdash \xi \leftrightarrow \zeta$. □

Then, for $\mathbf{N} \models T$, a model that makes it arithmetically sound,

Prop 3.29. If T is arithmetically sound, then for every $m \in \omega$ there is a sentence $\xi \in T$ such that $\mathbf{M} \models \xi$ while for no $k \leq n$ there is a Σ_k sentence in PA demonstrably equivalent to ξ.

Proof: The usual proof for PA is given in Rogers [54], p. 321. However we give here a slightly modified argument that imitates Proposition 3.24. First notice that

$$\emptyset^{(m+1)} = \{x : \phi_x^{\emptyset^{(m)}}(x)\}$$

is recursively enumerable but not recursive in $\emptyset^{(m)}$. Therefore, $\overline{\emptyset^{(m+1)}}$ isn't recursively enumerable in $\emptyset^{(m)}$, but contains a proper $\emptyset^{(m)}$–recursively enumerable set. Let's take a closer look at those sets.

We first need a lemma: form the theory $T^{(m+1)}$ whose axioms are those for T plus a denumerably infinite set of statements of the form "$n_0 \in \emptyset^{(n)}$,"

"$n_1 \in \emptyset^{(m)}$,"..., that describe $\emptyset^{(m)}$. Of course this theory doesn't have a recursively enumerable set of theorems. Then,

Lemma 3.30. *If $T^{(n+1)}$ is arithmetically sound, then $\phi_x^{\emptyset^{(m)}}(x) \downarrow$ if and only if*

$$T^{(m+1)} \vdash \exists x_1, \ldots, x_n \in \omega \, p(\rho(z), \langle z, d_{y,\emptyset^{(m)}} \rangle, x_1, \ldots, x_n) = 0.$$

Proof: Similar to the proof in the non–relativized case; see [48], p. 126 ff. □

Therefore we have that the oracle machines $\phi_x^{\emptyset^{(m)}}(x) \downarrow$ if and only if

$$T^{(m+1)} \vdash \exists x_1, \ldots, x_n \in \omega \, p(\rho(z), \langle z, d_{y,\emptyset^{(m)}} \rangle, x_1, \ldots, x_n) = 0.$$

However, since $\overline{\emptyset^{(m+1)}}$ isn't recursively enumerable in $\emptyset^{(m)}$ then there will be an index $m_0(\emptyset^{(m)}) = \langle \rho(z), \langle z, d_{y,\emptyset^{(m)}} \rangle \rangle$ such that

$$\mathbf{N} \models \forall x_1, \ldots, x_n [p(m_0, x_1, \ldots, x_n)]^2 > 0,$$

while it cannot be proved neither disproved within $T^{(m+1)}$. It is therefore demonstrably equivalent to a Π_{m+1} assertion. □

Now let $q(m_0(\emptyset^{(m)}), x_1, \ldots) = p(m_0(\emptyset^{(m)}), x_1, \ldots))^2$ be as in Proposition 3.29. Then:

Corollary 3.31. *If T is arithmetically sound, then for:*

$$\beta^{(m+1)} = \sigma(G(m_0(\emptyset^{(n)}))),$$

$$G(m_0(\emptyset^{(n)})) = \int_{-\infty}^{+\infty} \frac{C(m_0(\emptyset^{(n)}), x) e^{-x^2}}{1 + C(m_0(\emptyset^{(n)}), x)} dx,$$

$$C(m_0(\emptyset^{(n)}), x) = \lambda q(m_0(\emptyset^{(n)}), x_1, \ldots, x_r),$$

$\mathbf{N} \models \beta^{(m+1)} = 0$ *but for all* $n \leq m+1$, $\neg\{T^{(n)} \vdash \beta^{(m+1)} = 0\}$ *and* $\neg\{T^{(n)} \vdash \neg(\beta^{(m+1)} = 0)\}$. □

We have used here a variant of the construction of θ and β which first appeared in [16]. Then,

Corollary 3.32. *If T is arithmetically sound and if $\mathcal{L}_T$ contains expressions for the $\theta^{(m)}$ functions as given in Proposition 3.21, then for any nontrivial arithmetical predicate P there is a $\zeta \in \mathcal{L}_T$ such that the assertion $P(\zeta)$ is T–demonstrably equivalent to and T–arithmetically expressible*

as a Π_{m+1} assertion, but not equivalent to and expressible as any assertion with a lower rank in the arithmetic hierarchy.

Proof: As in the proof of Proposition 3.25, we write:

$$\zeta = \xi + [1 \stackrel{\bullet}{-} (p(m_0(\emptyset^m), x_1, \ldots, x_n) + 1)]\nu,$$

where $p(\ldots)$ is as in Proposition 3.29. □

Remark 3.33. Rogers discusses the rank within the arithmetical hierarchy of well–known open mathematical problems ([54], p. 322), such as Fermat's Conjecture—which in its usual formulation is demonstrably equivalent to a Π_1^0 problem,[6] or unsettled questions such as Riemann's Hypothesis, which is also stated as a Π_1^0 problem. On the other hand, the $P < NP$ hypothesis in computer science is formulated as a Π_2^0 sentence that can be made equivalent to an intuitive Π_1^0 sentence, while its negation, the $P = NP$ conjecture, can be formalized as a Π_1^0 sentence within Peano Arithmetic [19].

Rogers conjectures that our mathematical imagination cannot handle more that four or five alternations of quantifiers. However the preceding result shows that *any* arithmetical nontrivial property within T can give rise to intractable problems of arbitrarily high rank.

We stress the need for the extension $T \supset$ PA, since otherwise we wouldn't be able to find an expression for the characteristic function of a set with a high rank in the arithmetical hierarchy within our formal language. □

An extension of the preceding result is:

Corollary 3.34. *If T is arithmetically sound then, for any nontrivial P there is a $\zeta \in \mathcal{L}_T$ such that $P(\zeta)$ is arithmetically expressible, $\mathbf{N} \models P(\zeta)$ but only demonstrably equivalent to a Π_{n+1}^0 assertion and not to a lower one in the hierarchy.*

Proof: Put

$$\zeta = \xi + \beta^{(m+1)}\nu,$$

where one uses Corollary 3.31. □

[6]The question of whether Wiles' proof can be fully formalized within ZFC is still open, and so, while we know that Fermat's Theorem is true of the standard integers, we don't know which minimum axiomatic resources are required for its proof.

Beyond arithmetic

We recall:

Definition 3.35.

$$\emptyset^{(\omega)} = \{\langle x, y\rangle : x \in \emptyset^{(y)}\},$$

for $x, y \in \omega$. □

Then:

Definition 3.36.

$$\theta^{(\omega)}(m) = c_{\emptyset^{(\omega)}}(m),$$

where $c_{\emptyset^{(\omega)}}(m)$ is obtained as in Proposition 3.17. □

Still,

Definition 3.37.

$$\emptyset^{(\omega+1)} = (\emptyset^{(\omega)})'.$$

□

Corollary 3.38. $\mathbf{0}^{(\omega+1)}$ *is the degree of* $\emptyset^{(\omega+1)}$. □

Corollary 3.39. $\theta^{(\omega+1)}(m)$ *is the characteristic function of a nonarithmetic subset of* ω *of degree* $\mathbf{0}^{(\omega+1)}$. □

Corollary 3.40. *If* T *is arithmetically sound, then for:*

$$\beta^{(\omega+1)} = \sigma(G(m_0(\emptyset^{(\omega)})),$$

$$G(m_0(\emptyset^{(\omega)})) = \int_{-\infty}^{+\infty} \frac{C(m_0(\emptyset^{(\omega)}), x)e^{-x^2}}{1 + C(m_0(\emptyset^{(\omega)}), x)}dx,$$

$$C(m_0(\emptyset^{(\omega)}), x) = \lambda q(m_0(\emptyset^{(\omega)}), x_1, \ldots, x_r),$$

$\mathbf{N} \models \beta^{(\omega+1)} = 0$ *but* $T \not\vdash \beta^{(\omega+1)} = 0$ *and* $T \not\vdash \neg(\beta^{(\omega+1)} = 0)$. □

Prop 3.41. If T is arithmetically sound then given any nontrivial predicate P:

(1) There is a family of terms $\zeta_m \in \mathcal{L}_T$ such that there is no general algorithm to check, for every $m \in \omega$, whether or not $P(\zeta_m)$.

(2) There is a term $\zeta \in \mathcal{L}_T$ such that $\mathbf{M} \models P(\zeta)$ while $T \nvdash P(\zeta)$ and $T \nvdash \neg P(\zeta)$.
(3) Neither the ζ_m nor ζ are arithmetically expressible.

Proof: We take:

(1) $\zeta_m = x\theta^{(\omega+1)}(m) + (1 - \theta^{(\omega+1)}(m))y$.
(2) $\zeta = x + y\beta^{(\omega+1)}$.
(3) Neither $\theta^{(\omega+1)}(m)$ nor $\beta^{(\omega+1)}$ are arithmetically expressible. □

Remark 3.42. We have thus produced out of every nontrivial predicate in T intractable problems that cannot be reduced to arithmetic problems. Actually there are infinitely many such problems for every ordinal α, as we ascend the set of infinite ordinals in T. Also, the general nonarithmetic undecidable statement $P(\zeta)$ has been obtained without the help of any kind of forcing construction. □

For the way one proceeds with those extensions we refer the reader to references on the hyperarithmetical hierarchy [4, 22, 54].

3.10. First applications: mechanics and chaos theory

The search for some algorithmic procedure or at least for some reasonable criterion that would distinguish chaotic systems from non–chaotic ones was the original motivation that led to the results presented in this work. After striving for a short time to get one such criterion, the authors wondered around 1985 whether the question wasn't in fact algorithmically undecidable, perhaps due to the complexity of the behavior of systems that exhibit chaotic behavior, even if such systems are described by rather simple systems of equations.

We later saw that the metamathematical phenomena we were looking for had a different origin.

The original intuition was, roughly, that if a simple description encapsulates an involved behavior, then we would perhaps lack the tools to check for specific properties of the system like chaos, since the system's complexity might — perhaps — exceed by far the available tools for its analysis, whatever that might mean. However as it turned out the undecidability and incompleteness of chaotic dynamical systems turned out to stem from a totally different aspect of the question: as we have shown in detail, it is essentially a *linguistic* phenomenon, that is, it depends on the tools that we

have within formal systems to handle expressions for the objects in those systems.

(When we say "the undecidability and incompleteness of dynamical systems," we are making an abuse of language: more precisely, we mean the undecidability or incompleteness properties of the formal theory of those systems developed with the help of the language of classical analysis.)

Undecidability in classical mechanics

Let's go straight to the point and ask three questions in order to give a good example:

(1) Given a Hamiltonian h, do we have an algorithm that tells us whether the associated Hamiltonian dynamical system X_h can be integrated by quadratures?
(2) Given a Hamiltonian h such that X_h can be integrated by quadratures, can we algorithmically find a canonical transformation that will do the trick?
(3) Can we algorithmically check whether an arbitrary set of functions is a set of first integrals for a Hamiltonian system?

The answer to those questions is, no. Proof follows from the techniques developed in the previous sections [16].

Chaos theory is undecidable and incomplete

We finally reach the question that originally motivated our quest. Let X be a smooth vectorfield on a differentiable manifold M. Can we algorithmically check whether X has some kind of chaotic behavior — in any of the usual meanings for that word, that is, given an arbitrary vectorfield X, can we algorithmically decide whether X is chaotic?

This problem was explicitly discussed by M. Hirsch [35] when he makes some remarks about the Lorenz system of equations [43]:

> (...) By computer simulation Lorenz found that trajectories seem to wander back and forth between two particular stationary states, in a random, unpredictable way. Trajectories which start out very close together eventually diverge, with no relationship between long run behaviors.
>
> But this type of chaotic behavior has not been proved. As far as I am aware, practically nothing has been proved about this particular system (...)

> *A major challenge to mathematicians is to determine which dynamical systems are chaotic and which are not. Ideally one should be able to tell from the form of the differential equations.*

Emphasis is ours. Therefore we can ask:

> *Is there a general algorithmic criterion so that, if we specify some formal definition for chaos in a dynamical system, we can determine whether an arbitrary expression for a dynamical system satisfies that definition?*

Again the answer is, no.

Let M be a differentiable manifold, and let $U \subset M$ be a starshaped open domain. As always, we suppose that ZFC is arithmetically sound, and that our results happen within ZFC.

Remark 3.43. For the next proposition we need two specific results:

- The construction of a Hamiltonian system with a Smale horseshoe [36].
- The fact that some geodesic flows are Bernouilli flows.
  That is, those geodesic flows have a decidedly random behavior.

We assert:

Prop 3.44. There is no general algorithmic procedure to check:

(1) Whether an arbitrary vectorfield X over U is ergodic.
(2) If $\dim M \geq 4$, whether an arbitrary vectorfield X over U has a Smale horseshoe.
(3) If M is compact, real, two–dimensional, of class C^3 and has a constant negative curvature, whether an arbitrary X is a Bernouillian flow.

Proof: As above. For the first two assertions, let K_0 be a constant vectorfield on U, and let Y be ergodic, or have a Smale horseshoe (for an explicit example, see [36]). Now let θ be the "yes–no" function, or Halting Function. Then

$$Z_m = \Theta_m K_0 + (1 - \Theta_m)Y,$$

where

$$\Theta_m(x_1, \ldots, x_n) = \theta_m(x_1),$$

has the same smoothness properties of Y, as both K_0 and θ are constant functions. It is an undecidable, countable family of vectorfields, as in the preceding results.

For the third assertion, we know that geodesic flows on such an M are Bernouillian. Then, if X is one such flow, we write

$$Z_m = \theta(m)X.$$

Again we cannot in general decide whether we have a trivial zero field or a Bernouillian flow. □

We conclude with an incompleteness theorem:

Prop 3.45. If T contains an axiomatization of dynamical system theory then there is an expression X in the language of T so that

$$T \vdash T \text{ is a dynamical system}$$

while

$$T \nvdash X \text{ is chaotic}$$

and

$$T \nvdash \neg(X \text{ is chaotic}),$$

for any nontrivial predicate in the language of T that formally characterizes chaotic dynamical systems. □

That is to say, chaos theory is undecidable and incomplete (in its axiomatic version), no matter what nontrivial definition that we get for chaos.

3.11. Janus–faced physics

Theoretical physics has two faces. On one side, it allows us to do computations, to quantitatively obtain data that describe and predict the behavior of real–world systems. On the other side it allows us to imagine the inner workings of the phenomena out of the mathematical tools used in their description. This is the 'conceptual game' we mentioned before; we believe that it clarifies and complements Chaitin's vision of physical theories as algorithmic devices. The plethora of incompleteness results we've offered is of course a parallel to his vision of randomness as deeply ingrained into mathematics.

Let's consider one aspect of our work as an example. We have axiomatized physics with axiom systems where the dynamical rule is given by Dirac–like equations, instead of the more commonplace variational principles. Dirac–like equations are today an important tool in differential

geometry [57], where they are used in the classification of bundles over 4–dimensional manifolds (and 4–dimensional differential manifolds are our current formal depiction of spacetimes). They appear in K–theory and in modern theories of cohomology. When one uses Dirac–like equations in the axiomatization of physical theories one wishes to stress the wide–ranging relations that such a mathematical object has within today's geometry. Dirac–like equations are a kind of crossroad–concept in today's mathematics. They lead to manifold routes, many of them still unexplored.

We've briefly mentioned 4–dimensional differential manifolds. The problem of the physical meaning of exotic, "fake" spacetimes, if any, is still wide open. We've, again briefly, mentioned it before [15], when we pointed out that there is an extra difficulty in that question: once we have uncountable many exotic differentiable structures for some fixed adequate topological 4–manifold, we will have uncountable many *set theoretically generic* differentiable structures for that manifold in adequate models for our theory. What is their meaning?

We don't know. Our axiomatization for classical and first–quantized physics opens up large vistas towards unknown, totally new, landscapes. Such is its *raison d'être.*

Acknowledgments

The authors wish to thank their institutions for support. N. C. A. da Costa acknowledges support from CNPq, Philosophy Section; F. A. Doria thanks C. A. Cosenza and S. Fuchs for their invitation to join respectively the Fuzzy Sets Research Group and the Philosophy of Science Group at the Production Engineering Program, COPPE–UFRJ.

References

[1] R. Abraham, J. Marsden, *Foundations of Mechanics*, 2nd. ed., Addison–Wesley (1978).

[2] S. Albeverio, J. E. Fenstad, R. Högh–Krohn, T. Lindström, *Nonstandard Methods in Stochastic Analysis and Mathematical Physics*, Academic (1986).

[3] V. I. Arnol'd, *Les Méthodes Mathématiques de la Mécanique Classique*, Mir, Moscow (1976).

[4] C. J. Ash and J. Knight, *Computable Structures and the Hyperarithmetical Hierarchy*, Elsevier (2000).

[5] M. F. Atiyah, *Geometry of Yang–Mills Fields*, Lezioni Fermiane, Pisa (1979).

[6] P. A. Benioff, "Models of Zermelo Frankel set theory as carriers for the mathematics of physics. I," *J. Math. Phys.* **32**, 618 (1976).
[7] P. A. Benioff, "Models of Zermelo Frankel set theory as carriers for the mathematics of physics. II," *J. Math. Phys.* **32**, 629 (1976).
[8] N. Bourbaki, *Set Theory*, Hermann and Addison–Wesley (1968).
[9] R. Carnap, *The Logical Syntax of Language*, Routledge and Kegan Paul (1949).
[10] R. Carnap, *Introduction to Symbolic Logic and its Applications*, Dover (1958).
[11] Y. M. Cho, "Higher–dimensional unifications of gravitation and gauge theories," *J. Math. Physics* **16**, 2029 (1975).
[12] L. Corry, "David Hilbert and the axiomatization of physics (1894–1905)," *Arch. Hist. Exact Sciences* **51**, 83 (1997).
[13] E. M. Corson, *Introduction to Tensors, Spinors and Relativistic Wave–Equations*, Blackie & Sons. (1953).
[14] N. C. A. da Costa and R. Chuaqui, "On Suppes' set–theoretical predicates," *Erkenntnis* **29** 95 (1988).
[15] N. C. A. da Costa and F. A. Doria, "A Suppes predicate for general relativity and set–theoretically generic spacetimes," *Int. J. Theoretical Phys.* **29**, 935 (1990).
[16] N. C. A. da Costa and F. A. Doria, "Undecidability and incompleteness in classical mechanics," *Int. J. Theoretical Physics* **30**, 1041 (1991).
[17] N. C. A. da Costa and F. A. Doria, "Suppes predicates and the construction of unsolvable Problems in the axiomatized sciences," P. Humphreys, ed., *Patrick Suppes, Scientific Philosopher*, II, 151–191 Kluwer (1994).
[18] N. C. A. da Costa and F. A. Doria, "Computing the Future," in K. V. Velupillai, ed., *Computability, Complexity and Constructivity in Economic Analysis*, Blackwell (2005).
[19] N. C. A. da Costa, F. A. Doria and E. Bir, "On the metamathematics of *P vs. NP*," to appear in *Appl. Math. Computation* (2007).
[20] N. C. A. da Costa and S. French, *Science and Partial Truth*, Oxford (2003).
[21] M. Davis, "Hilbert's Tenth Problem is unsolvable," *Amer. Math. Monthly* **80**, 233 (1973).
[22] M. Davis, *Computability and Unsolvability*, Dover (1982).
[23] P. A. M. Dirac, *The Principles of Quantum Mechanics*, Oxford U. P. (1967).
[24] F. A. Doria, "A Dirac–like equation for the gravitational field," *Lett Nuovo Cimento* **14**, 480 (1975).
[25] F. A. Doria, "A Lagrangian formulation for noninteracting high–spin fields," *J. Math. Phys.* **18**, 564 (1977).
[26] F. A. Doria, "Informal and formal mathematics," to appear in *Synthèse* (2007).
[27] F. A. Doria, A. F. Furtado do Amaral, S. M. Abrahão, "A Dirac–like equation for gauge fields," *Progr. theor. Phys* **75**, 1440 (1986).
[28] F. A. Doria and J. F. Costa, eds., Special issue on hypercomputation, *Applied Math. Computation* **178** (2006).
[29] A. Einstein, *The Meaning of Relativity*, Methuen (1967).

[30] A. Einstein and others, *The Principle of Relativity*, Dover s/d.
[31] G. Emch, *Algebraic Methods in Statistical Mechanics and Quantum Field Theory*, Wiley (1972).
[32] P. R. Halmos, *Naive Set Theory*, Van Nostrand (1960).
[33] H. Hertz, *The Principles of Mechanics*, transl. by D. E. Jones and J. T. Walley, Dover (1956).
[34] D. Hilbert, "Mathematical Problems," in F. E. Browder, ed., *Mathematical Developments Arising from the Hilbert Problems,* Proc. Symp. Pure Math. **28**, AMS (1976).
[35] M. Hirsch, "The Chaos of Dynamical Systems," in P. Fischer and W. R. Smith, eds., *Chaos, Fractals and Dynamics*, Marcel Dekker (1985).
[36] P. J. Holmes and J. Marsden, "Horseshoes in perturbations of Hamiltonian systems with 2 degrees of freedom," *Commun. Math. Phys.* **82**, 523 (1982).
[37] J. D. Jackson, *Classical Electrodynamics*, John Wiley (1962).
[38] J. H. Jeans, *Mathematical Theory of Electricity and Magnetism*, Cambridge U. P. (1925).
[39] T. Jech, *Set Theory*, Academic Press (1978).
[40] F. W. Kamber and P. Tondeur, *Foliated Bundles and Characteristic Classes*, Lecture Notes in Mathematics # 493, Springer (1975).
[41] G. T. Kneebone, *Mathematical Logic*, Van Nostrand (1963).
[42] C. Lanczos, *The Variational Principles of Mechanics*, U. of Toronto Press (1977).
[43] E. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.* **20**, 130 (1963).
[44] A. J. Lotka, *Elements of Mathematical Biology*, Dover (1956).
[45] G. Ludwig, *Les Structures de Base d'une Théorie Physique*, Springer (1990).
[46] C. A. Lungarzo, "Superposition of States in Quantum Logic from a Set–Theoretical Point of View," in A. I. Arruda, N. C. A. da Costa and R. Chuaqui, eds., *Mathematical Logic—Proceedings of the First Brazilian Conference*, Lecture Notes in Pure and Applied Math., Marcel Dekker (1976).
[47] S. MacLane, *Geometrical Mechanics*, monograph, Dept. of Mathematics, U. of Chicago (1968).
[48] M. Machtey and P. Young, *An Introduction to the General Theory of Algorithms*, North–Holland (1979).
[49] J. D. Maitland Wright, "All operators on Hilbert space are bounded," *Bull. of the American Math. Soc.* **79**, 1247 (1973).
[50] J. C. Maxwell, *A Treatise on Electricity and Magnetism*, I and II, Dover (1954).
[51] E. Mendelson, *Introduction to Mathematical Logic*, 4th ed., Chapman & Hall (1997).
[52] A. O'Rahilly, *Electromagnetic Theory*, I and II, Dover (1965).
[53] E. Post, "Recursively enumerable sets of positive integers and their decision problems," *Bull. AMS* **50**, 284 (1944).
[54] H. Rogers Jr., *Theory of Recursive Functions and Effective Computability*, McGraw–Hill (1967).
[55] B. Russell, *The Analysis of Matter*, Dover (1954).
[56] P. A. Samuelson, *Foundations of Economic Analysis*, Atheneum (1967).

[57] A. Scorpan, *The Wild World of 4–Manifolds*, AMS (2005).
[58] R. M. Solovay, "A model of set theory in which every set of reals is Lebesgue measurable," *Ann. Math,* **92**, 1 (1970).
[59] J. A. Stratton, *Electromagnetic Theory*, McGraw–Hill (1941).
[60] S. Sternberg, *Lectures on Differential Geometry*, Benjamin (1964).
[61] P. Suppes, *Introduction to Logic*, Van Nostrand (1957).
[62] P. Suppes, *Representation and Invariance of Scientific Structures*, CSLI, Stanford University (2002).
[63] M. A. Tonnelat, *Les Théories Unitaires de l'Électromagnétisme et de la Gravitation*, Gauthier–Villars (1965).
[64] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. London Math. Society* **50**, 230 (1937).
[65] R. Utiyama, "Invariant theoretical interpretation of interaction," *Phys. Review* **101**, 1597 (1956).
[66] B. L. van der Waerden, ed., *Sources of Quantum Mechanics*, Dover (1968).
[67] H. Weyl, *Space Time Matter*, Dover (1952).
[68] A. S. Wightman, "Hilbert's Sixth Problem: Mathematical Treatment of the Axioms of Physics," in F. E. Browder, ed., *Mathematical Developments Arising from the Hilbert Problems*, Proc. Symp. Pure Math. **28**, AMS (1976).

Chapter 4

The Implications of a Cosmological Information Bound for Complexity, Quantum Information and the Nature of Physical Law

P. C. W. Davies

BEYOND: Center for Fundamental Concepts in Science
Arizona State University, USA; `deepthought@asu.edu`

Whereof one cannot speak, thereof one must remain silent.

Ludwig Wittgenstein[1]

Is our universe a polynomial or an exponential place?

Scott Aaronson[2]

4.1. What are the laws of physics?

Gregory Chaitin is undoubtedly one of the most profound thinkers of our time. I have drawn on his work at several stages in my own career development, and especially in formulating the ideas that follow. It is an honor to contribute to this volume to celebrate Chaitin's important insights into mathematics, computing and physical reality.

I should like to start with a quotation from Chaitin's recent book *Meta-Math*: "Why should I believe in a real number if I can't calculate it, if I can't prove what its bits are, and if I can't even refer to it? ... The real line from 0 to 1 looks more and more like a Swiss cheese." [3] In other words, the real line is a useful fiction, an unattainable idealization. The question I wish to address here is how this sweeping conclusion impacts on my own field of theoretical physics and cosmology. The real line, its extension to the complex plane, together with the related properties of differentiability, play an absolutely central role in theoretical physics, on account of the fact that all the known fundamental laws of physics are expressed in terms of differentiable functions defined over the set of real or complex numbers. So I want to start by asking a very basic, but surprisingly little addressed

question: What are the laws of physics and where do they come from? The subsidiary question, Why do they have the form that they do? I have discussed in detail elsewhere.[4]

First let me articulate the orthodox position, adopted by most theoretical physicists, which is that the laws of physics are immutable, absolute, eternal, perfect mathematical relationships, infinitely precise in form. The laws were imprinted on the universe at the moment of creation, i.e. at the big bang, and have since remained fixed in both space and time. The properties of the physical universe depend in an obvious way on the laws of physics, but the basic laws themselves depend not one iota on what happens in the physical universe. There is thus a fundamental asymmetry: the states of the world are affected by the laws, but the laws are completely unaffected by the states – a dualism that goes back to the foundation of physics with Galileo and Newton. The ultimate source of the laws is left vague, but it is tacitly assumed to transcend the universe itself, i.e. to lie beyond the physical world, and therefore beyond the scope of scientific inquiry. The proper task of the physicist, it is often said, is to discover the forms of the laws using reason and experiment, adopt them pragmatically, and get on with the job of determining their consequences. Inquiry into their origin is discouraged as a quasi-religious quest.

The orthodox view of the nature of physical laws conforms well to the mathematical doctrine of Platonism. Plato regarded mathematical forms and relationships as enjoying a real existence in an otherworldly realm, where mathematicians come upon them in a voyage of intellectual discovery. A Platonist regards mathematics as possessing an existence independent of the physical universe, rather than being a product of the human brain. An essential quality of the Platonic heaven is that the mathematical forms it contains are perfect. For example, circles are exactly round, in contrast to circles in the physical universe, which are always flawed approximations to the idealized Platonic forms.

Most theoretical physicists are by temperament Platonists. They envisage the laws of physics too as perfect idealized mathematical relationships and operations that really exist, located in an abstract realm transcending the physical universe. I shall call this viewpoint physical Platonism to distinguish it from mathematical Platonism. Newton was a physical Platonist, and cast his laws of mechanics and gravitation in terms of what we would now call real numbers and differentiable functions. Taking Newton's laws seriously implies accepting infinite and infinitesimal quantities, and arbi-

trary precision. The idealized, Platonic notion of the laws of physics reached its zenith with the famous claim of Laplace, concerning an omniscient demon. Laplace pointed out that the states of a closed deterministic system, such as a finite collection of particles subject to the laws of Newtonian mechanics, are completely fixed once the initial conditions are specified[5].

"We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at any given moment knew all of the forces that animate nature and the mutual positions of the beings that compose it, if this intellect were vast enough to submit the data to analysis, could condense into a single formula the movement of the greatest bodies of the universe and that of the lightest atom; for such an intellect nothing could be uncertain and the future just like the past would be present before its eyes."

If Laplace's argument is taken seriously, on the assumptions adopted, then everything that happens in the universe, including Laplace's decision to write the above words, my decision to write this article, Chaitin's beautiful work on Omega, etc. are all preordained. The information about these events is already contained in the state of the universe at any previous time. To get some idea of the demon's gargantuan task, note the following. If the demon overlooked the gravitational force of a single electron located at the edge of the observable universe, then his prediction for the motion of a given molecule of air in your living room would be rendered completely uncertain after only 12 intermolecular collisions.[6] This arresting example reveals how exquisitely sensitive to error predicting the future can be. Laplace's vignette is based on classical mechanics, and is usually dismissed by invoking quantum mechanics, or arguing that the universe is an open system, but this misses the point. The real absurdity in Laplace's statement is its implicit reliance on physical Platonism extrapolated to a staggering degree, made without any experimental foundation whatever.In spite of the fact that we now know Newtonian mechanics is only an approximation, physical Platonism remains the dominant philosophy among theoretical physicists. The project of quantum cosmology, for example, is predicated on the assumption that the laws of quantum mechanics and general relativity exist independently of the universe, and may therefore be invoked to explain how the universe came to exist from nothing. In the fashionable subject of string/M theory, the string Lagrangian, or whatever else serves to determine the unified dynamics, is assumed to somehow "already exist", so that from it may (one day) flow an explanation for space,

time, matter and force.

4.2. Laws as software

A completely different view of the relationship between mathematics and physics comes from Chaitin's development of algorithmic information theory, from which he was drawn to the conclusion, "A scientific theory is like a computer program that predicts our observations."[7] For example, in Newtonian mechanics the initial positions and momenta of a system of particles serve as input data, the laws of mechanics are the program, and the final state of the particles at some later time of interest corresponds to the output. In this manner, the universe processes information automatically as it evolves. So we might envisage the laws of physics in terms of software, as a grand cosmic computer program. This shift of perspective, simple though it may be, has profound implications, which are immediately apparent when we ask what is the hardware on which the cosmic software is being run? The answer is, of course, the universe itself. And by this I mean the real, physical universe. I am not referring to some imaginary cosmic hardware in a Platonic heaven, but the real universe we observe. The significance of this last point is that the real universe might very well be finite, that is, have finite resources and age, and thus be subject to restrictions on what it can accomplish in regards to computation.

Why might the universe be finite in resources? What matters for computational purposes is not the spatial extent of the universe, but the number of physical degrees of freedom located in a causally connected region. Information processed in causally disconnected parts of space cannot be considered as belonging to the same "program." In the standard cosmological models, the region of the universe to which we have causal access at this time is limited by the finite speed of light and finite age of the universe (since the big bang). That is, there exists a "particle horizon" in space, measuring some billions of light years across at this time. The region within our particle horizon contains about 10^{80} particles of matter, and about 10^{90} photons and neutrinos. If the system is treated quantum mechanically, with information encoded in discrete bits (e.g. spin up, spin down), then the maximum number of bits of information contained in a horizon volume at this time is about 10^{122} according to Seth Lloyd.[8] His calculation takes into account the gravitational degrees of freedom too. If the universe is uniform, any other causal region would possess a similar

upper bound. Thus we may write

$$\mathrm{I}_{\mathrm{universe}} \leq 10^{122}. \tag{1}$$

The bound (1) is not fixed, but grows with time as the horizon expands and encompasses more particles:

$$\mathrm{I}_{\mathrm{universe}} \propto t^2. \tag{2}$$

It is a simple matter, using quantum mechanics and thermodynamics, to also calculate the maximum amount of information that could have been processed (i.e. the total number of possible bit flips) in our causal region since the origin of the universe. The answer comes out again of order 10^{122} , taking into account Eq. (2), i.e. the fact that the causal region was smaller in the past and so encompassed less particles.

A similar information bound may be derived from an entirely different line of argument, exploiting the link between physics and information discovered by Bekenstein[9] and Hawking[10] when applying quantum mechanics to black holes. They found that an uncharged, non-rotating black hole possesses entropy S given by

$$\mathrm{S} = 4\pi kGM^2/\hbar c^3 = \tfrac{1}{4}A, \tag{3}$$

where M and A are the mass and area of the black hole respectively, and the other symbols have their usual meanings as various fundamental constants of nature.

The fact that the entropy is a function of black hole *area*, as opposed to volume, is deeply significant. In the case of a laboratory gas, for example, entropy is additive: twice the volume of a (homogeneous) gas will have twice the entropy. Evidently, when gravitation enters the picture, the rules of the game change fundamentally. Entropy can been regarded as a measure of information I (or information loss), through the relationship

$$S = k \log_2 I \tag{4}$$

so the Bekenstein-Hawking formula (3) relates the total information content of a region of space to the area of the surface encompassing that volume. The information inside a black hole is lost because an observer in the external region cannot access it on account of the fact that the surface of the hole is an event horizon. (There remains an unresolved issue about whether the information is permanently lost, or just rendered inaccessible until the black hole eventually evaporates. I shall not consider that topic further in this chapter.) A useful way to think about Eq. (3) is to define the Planck

length $L_P \equiv (G/\hbar c^3)^{1/2}$ as a fundamental unit, and note that, using Eq. (4), the information of the black hole is simply one quarter of the horizon area in Planck units.

Early on, Bekenstein sought to generalize his result by postulating that Eq. (1) serves as a *universal* bound on entropy (or information content) applicable to *any* physical system.[11] That is, the information content of a physical system can never, he claims, exceed one quarter of the area of its encompassing surface. The black hole saturates the Bekenstein bound, and represents the maximum amount of information that can be packed into the volume occupied by the hole, as befits the equilibrium end state of a gravitating system. A simple argument in support of the universal Bekenstein bound is that if a system confined to a certain region of space possessed an information content in excess of the bound, one could then add some matter and induce this system to undergo gravitational collapse to a black hole, thereby reducing its entropy and violating the second law of thermodynamics (suitably generalized to include event horizon area). However, the Bekenstein bound remains a conjecture: a general proof is lacking.The idea of associating entropy and information with horizon area was soon extended to include *all* event horizons, not just those surrounding black holes. For example, if the universe becomes dominated by dark energy, which is what current astronomical observations suggest, it will continue to expand at an accelerating rate (dark energy acts as a sort of antigravity force). This creates a *cosmological* event horizon, which may be envisaged as a roughly spherical surface that bounds the region of the universe to which we can ever have causal and informational access. A similar horizon characterizes the period of inflation, widely believed to have occurred at about 10^{-34} s after the big bang. Generalizations of horizon entropy have been proposed for cosmological horizon area too, with de Sitter space (a universe subject to dark energy alone) saturating the Bekenstein bound, by Gibbons and Hawking[12], Bousso[13], and Davis and Davies[14]. A number of calculations support the proposal. Based on the foregoing ideas, 't Hooft[15] and Susskind[16] have proposed the so-called *holographic principle*, according to which the information content of the entire universe is captured by an enveloping surface that surrounds it. The principle states that the total information content of a region of space cannot exceed one quarter of the surface area that confines it (other variants of the holographic principle have been proposed, with different definitions of the enveloping area), and that this limit is attained in the case of the cosmological event horizon. A

simple calculation of the size of our universe's event horizon today based on the size of the event horizon created by the measured value of dark energy gives an information bound of 10^{122} bits, the same as found by Lloyd using the particle horizon. The event horizon also expands with time, and at this epoch is roughly the same radius as the particle horizon, but unlike the latter, it asymptotes to a constant value not a lot greater than its present value (assuming that the density of dark energy is constant). So whether we take the particle horizon or the event horizon, or a more generalized holographic principle, as the basis for the calculation, we discover an upper bound like (1) on the information content of a causal region of the universe.

How might the bound affect physics and cosmology? The answer to this question depends critically on one's assumptions about the nature of information. The traditional logical dependence of laws, states of matter and information is

A. laws of physics → matter → information.

Thus, conventionally, the laws of physics form the absolute and eternal bedrock of physical reality and, as mentioned, cannot be changed by anything that happens in the universe. Matter conforms to the "given" laws, while information is a derived, or secondary property having to do with certain special states of matter. But several physicists have suggested that the logical dependence should really be as follows:

B. laws of physics → information → matter.

In this scheme, often described informally by the dictum "the universe is a computer," information is placed at a more fundamental level than matter. Nature is regarded as a vast information-processing system, and particles of matter are treated as special states which, when interrogated by, say, a particle detector, extract or process the underlying quantum state information so as to yield particle-like results. It is an inversion famously encapsulated by Wheeler's pithy phrase 'It from bit'.[17] Treating the universe as a computer has been advocated by Fredkin[18], Lloyd[8] and Wolfram[19] among others. An even more radical transformation is to place *information* at the base of the logical sequence, thus

C. information → laws of physics → matter.

The attraction of scheme C is that, after all, the laws of physics are informational statements.

For most purposes the order of logical dependence does not matter much, but when it comes to the information bound on the universe, one is forced to confront the status of information: is it ontological or epistemological? If information is simply a description of *what we know* about the physical world, as is implied by Scheme A, there is no reason why Mother Nature should care about the limit (1). Or, to switch metaphors, the bedrock of physical reality according to Scheme A is sought in the perfect laws of physics, which live elsewhere, in the realm of the gods – the Platonic domain they are held by tradition to inhabit – where Mother Nature can compute to arbitrary precision with the unlimited quantity of information at her disposal. According to orthodoxy, the Platonic realm is the "real reality," while the world of information is but the shadow on Plato's cave. But if *information* underpins physical reality – if, so to speak, it occupies the ontological basement – (as is implied in Scheme C and perhaps B) then the bound on I_{universe} represents a fundamental limitation on *all reality*, not merely on states of the world that humans perceive.

Someone who advocated precisely this latter position was Rolf Landauer, a former colleague of Chaitin's at IBM. He explicitly took the view that "the universe computes in the universe," because he believed, as he was fond of declaring, that "information is physical." And Landauer was quick to spot the momentous consequences of this shift in perspective:

"The calculative process, just like the measurement process, is subject to some limitations. A sensible theory of physics must respect these limitations, and should not invoke calculative routines that in fact cannot be carried out."[20]

In other words, in a universe limited in resources and time – a universe subject to the information bound (1) in fact – concepts like real numbers, infinitely precise parameter values, differentiable functions, the unitary evolution of a wave function – are a fiction: a useful fiction to be sure, but a fiction nevertheless, and with the potential to mislead. It then follows that the laws of physics, cast as idealized infinitely precise mathematical relationships inhabiting a Platonic heaven, are also a fiction when it comes to applications to the real universe. Landauer's proposal that our theories should be constrained by the – possibly finite – resources of the universe has been independently developed in recent years by Benioff.[21]

If one adopts Landauer's philosophy, then some serious consequences follow. In effect, one cannot justify the application of the laws of physics in

situations employing calculations that involve numbers greater than about 10^{122}, and if one does, then one might expect to encounter departures between theory and experiment. What might this mean in practice? Well, for many purposes, the information bound is so large that the consequences are negligible. Take for example the law of conservation of electric charge. If this were to fail at the 10^{122} bit level of accuracy, the implications are hardly dire. (The law has been tested only to about one part in 10^{12}.)

There are situations in which very large numbers routinely crop up in theoretical physics calculations. One obvious class of cases is where exponentiation occurs. Consider, for example, statistical mechanics, where Poincaré recurrence times are predicted to be of order $\exp(10^N)$ Planck times (chosen to make the number dimensionless) and N is the number of particles in the system. Imposing a bound of 10^{122} implies that the recurrence time prediction is reliable only for recurrence times of about 10^{60} years. Again, this is so long we would be unlikely to notice any departure between theory and observation. Closely related is the problem of Laplace's demon already discussed. Imposing the information bound renders the demonic prediction valueless almost immediately, because the bound will be exhausted after of order one bit-flip of the 10^{122} degrees of freedom in the universe. Exponentiation arises in chaos theory too, via the Lyapunov coefficient. In these examples, the fact that the underlying deterministic mechanics might possess only finite precision is of little importance, because any uncertainties thereby generated are already totally swamped by the practical breakdown of predictability involved in complex and/or chaotic systems.

A case of exponentiation in a relatively simple system occurs in general relativity in connection with the formation of event horizons. For example, when a star implodes to form a black hole, light leaving the surface of the star is exponentially redshifted with an e folding time typically of order a few microseconds. What happens, then, if the exponential redshift is cut off at 10^{122} Planck lengths? The classical properties of the black hole are scarcely affected, but Hawking's original derivation of black hole radiance is invalidated, as it already well known.[22] Inflation in the very early universe involves an exponential rate of expansion, i.e. a de Sitter phase, and this offers a stringent test of the information bound hypothesis. It is a key feature of the information bound that it is time-dependent. In the past, the bound was smaller, and its effects on physics would have been greater (see Eq.(2)). During the very early universe, the effects could have been

significant, and may have left a trace on the structure of the universe that could be used to test the existence of the bound. Inflation is a brief episode of exponential expansion thought to have occurred at about 10^{-34} s after the big bang. At that time, the horizon size was about 3×10^{-24} cm, yielding a surface of about 10^{-19} Planck areas. The information bound then implies for the cosmological scale factor change

$$a(t_{after})/a(t_{before}) < 10^{19}. \tag{5}$$

Guth's original proposal was for an inflation factor at least 10^{20}, so (given the rough-and-ready nature of the calculation) the information bound is consistent with inflation, but only marginally so, and a more detailed analysis may suggest observable consequences, such as a measurable departure from spatial flatness

Another class of problems in which large numbers are unavoidable is quantum mechanics and quantum field theory, and it is to that topic that I now turn.

4.3. The quantum vacuum

In quantum mechanics the state of the system is described by a vector in a Hilbert space. For a generic problem, the Hilbert space will possess an infinite number of dimensions. Clearly this construction comes into conflict with the information bound hypothesis. A simple example of the problem concerns the energy of the quantum vacuum, evaluated by summing zero point modes over an infinite set of simple harmonic oscillators.[23] For a massless scalar field confined to a cube of space of linear dimension L, the energy density ρ of the vacuum is given by

$$\rho = \tfrac{1}{2}\hbar c L^{-1} \textstyle\sum_k \omega, \tag{5}$$

where the sum is taken over all the field modes of momentum k. The right hand side of Eq. (5) diverges like $\sim \omega^4$ as $\omega \to \infty$. It may be rendered finite by imposing a cut-off in the summation. A natural cut-off is provided by the Planck frequency, which incorporates only the fundamental constants already present in the theory: $\hbar$, c and G. Using this cut-off, Eq. (5) yields a vacuum energy density of 10^{113} Jm^{-3}, which is some 10^{122} times the observed dark energy density. This staggering discrepancy between theory and observation has been known for many years, and is known as the dark energy (or cosmological constant) problem. It is one of the main outstanding challenges to physical theory.

The occurrence of the same factor 10^{122} in this discrepancy as in the cosmological information bound is a clear pointer to an alternative explanation for dark energy, and indeed, inequality (1) provides a second natural cut-off for the summation in Eq. (5). Rewriting (5) in terms of modes,

$$\rho \approx \hbar c L^{-4} \sum n^4. \tag{6}$$

If it is now argued that the sum Σn^4 should be bounded by (1), then taking L to be the horizon radius (roughly a Hubble radius) and $\Sigma n^4 \sim 10^{122}$, we may evaluate the vacuum energy density to be

$$\rho \approx 10^{-9}\, Jm^{-3} \approx \rho_{observed}. \tag{7}$$

The same result may be derived in a completely different way, by imposing the condition on the vacuum energy that at every scale of size L, the energy density must not exceed the level at which the total mass within a volume L^3 is greater than the mass of a black hole of size L, otherwise the vacuum energy would presumably undergo gravitational collapse. This requirement may be expressed as follows:

$$\rho \mathrm{c}^2 L^3 < M_{bh}(L). \tag{8}$$

Substituting the right hand side of Eq. (5) for ρ we obtain, to an order or magnitude,

$$G\hbar\omega^4 L^3/c^7 < L \tag{9}$$

or

$$\rho < c^4/GL^2 \tag{10}$$

Taking L to be the Hubble radius, inequality (10) may be re-cast in the following suggestive form[24]:

$$\rho < (\rho_P \rho_H)^{1/2} \approx 10^{-9}\, Jm^{-3} \approx \rho_{observed} \tag{11}$$

where ρ_P is the Planck energy density and ρ_H is the Hubble energy density, defined to be the energy density of a single quantum in a Hubble volume with a wavelength equal to the Hubble radius.

This remarkable result – that the cosmological information bound explains the magnitude of the dark energy – comes at a price, however. The same reasoning may be applied to the *pressure* of the vacuum, p, which for a massless scalar field is

$$p = -\tfrac{1}{2}\hbar \mathrm{c} \mathrm{L}^{-1} \Sigma\, \omega \tag{12}$$

i.e. $p = -\rho$, which is the necessary equation of state for the vacuum energy to play the role of dark energy. Now recall that the information bound varies with time in the manner indicated by Eq. (2). Hence the cut-off in the summation in both Eqs. (5) and (12) will be time-dependent, so the dark energy is also predicted to be time-dependent. This raises an immediate difficulty with the law of energy conservation:

$$pda^3 + d(\rho a^3) = 0 \tag{13}$$

which can be satisfied for a time-dependent p and ρ only if there is some compensatory change, e.g. G and/or c vary with time. There is a substantial literature on such holographic cosmological models,[25] including comparison with observations, which I shall not review here.

4.4. Quantum information processing

As a final application of the information bound hypothesis, let me turn to non-relativistic quantum mechanics. A transformation in our understanding of information came with the recognition that because nature is fundamentally quantum mechanical, the rules for information processing at the quantum level differ not only in the technical details but in their very conceptual basis from the classical case. In conventional (classical) information theory, the basic unit is the bit, or binary choice, usually symbolized by 0 and 1. In quantum mechanics, the bit is replaced by a more abstract entity: the qubit. When humans read out the information content of a quantum system, they appropriate only bits – the act of read-out collapses qubits into bits. But the importance of quantum information dynamics is that in an isolated unobserved quantum system, the qubits generally evolve in a manner completely different from the classical case, involving the whole panoply of quantum weirdness, including, most crucially, superposition and entanglement. It is this feature that has commended quantum information science to governments and business by holding out the promise of large-scale quantum computation. By exploiting qubit dynamics, a quantum computer would represent an unprecedented leap in computational power.[26]

The key to quantum computation lies with the exponential character of quantum states. Whereas a classical binary switch is either on (1) or off (0), a quantum system can be in a superposition of the two. Furthermore, a multi-component quantum system can incorporate entanglement of spatially separated subsystems. Combining these two properties implies that an n-component system (e.g. n atoms) can have 2^n states, or components

of the wave function, that describe the system. If it were possible to control all the components, or branches, of the wave function simultaneously, then the quantum system would be able to process information exponentially more powerfully than a classical computer. This is the aspiration of the quantum computation project.

Because the complexity of an entangled state rises exponentially with the number of qubits (which is its virtue), large-scale quantum information processing comes into conflict with the information bound. Specifically, a quantum state with more components than about $n = \log_2 I_{\rm universe}$ will require more bits of information to specify it than can be accommodated in the entire observable universe! Using the bound given by inequality (1), this yields a limit of approximately $n = 400$. In other words, a generic entangled state of more than about 400 particles will have a quantum state with more components than $I_{\rm universe}$, evolving in a Hilbert space with more dimensions than $I_{\rm universe}$. The question therefore arises of whether this violation of the information bound (1) signals a fundamental physical limit. It seems to me that it must.

On the face of it, the limit of 400 particles is stringent enough to challenge the quantum computation industry, in which a long-term objective is to entangle many thousands or even millions of particles and control the evolution of the quantum state to high precision. The foregoing analysis, however, is overly simplistic. First, note that the dimensionality of the (non-redundant part of the) Hilbert space is not an invariant number: by changing the basis, the number might be reduced. So specifying the complexity of a quantum state simply by using the dimensionality of the Hilbert space can be misleading. A more relevant criterion is the number of independent parameters needed to specify inequivalent n-component quantum systems. This problem has been addressed, but it is a difficult one on which only limited progress has so far been made.[27] Second, the dimensionality of the Hilbert space serves to define the number of amplitudes needed to specify a generic superposition. But the amplitudes themselves require additional information to specify them; indeed, a single complex number coefficient α_i will mostly contain an infinite number of bits of information. If we are to take the bound (1) seriously, then it must be applied to the total algorithmic information content of the amplitude set over the entire Hilbert space. Following Chaitin, the algorithmic information measure of a binary string X is defined as

$$H(X) = -\ln P(X) + \mathrm{O}(1) \tag{14}$$

where $P(X)$ is the probability that the proverbial monkey typing randomly on a typewriter will generate a program which, when run on a universal Turing machine, will output X. Applied to the amplitude set $\{\alpha_i\}$ of a generic quantum state (plus any ancillary information needed to specify the state, such as constraints), the cosmological information bound (1) may be expressed as follows:

$$H(\{\alpha_i\}) < A_{holo}/L_P^2 \tag{15}$$

where A_{holo} is the area of the appropriate holographic surface (e.g. a cosmological event horizon). Inequality (17) is a stronger constraint than (1), appropriate to the interpretation of information as ontological and fundamental, and therefore including not merely a head-count of the degrees of freedom, but the algorithmic information content of all the specifying parameters of the state too. This extra informational burden on the bound will reduce somewhat the dimensionality of the Hilbert space at which unitary evolution is expected to break down.

A more subtle issue concerns the specific objectives of quantum computation, which is not to control the dynamical evolution of *arbitrary* entangled quantum states, but an infinitesimal subset associated with certain mathematical problems of interest, such as factoring. It is trivially true that it is impossible to prepare, even approximately, a state containing more than 10^{122} truly independent parameters because it is impossible to even specify such a state: there are not enough bits in the universe to contain the specification. Almost all states fall into this category of being impossible to specify, prepare and control. So in this elementary sense, generic quantum computation is obviously impossible. Less obvious, however, is whether the subset of states (of measure zero) of interest to the computing industry is affected by the cosmological information bound, for even if it is the case that the number of independent amplitudes exceeds 10^{122}, there may exist a compact mathematical algorithm to generate those amplitudes. (The algorithm for generating the amplitudes that specify the initial state should not be confused with the algorithm to be executed by the quantum computer dynamics.) For example, the amplitudes of the quantum computer's initial state could be the (unending) digits of π, which can be generated by a short algorithm. That is, the set of amplitudes may contain an unbounded number of bits of information, but a finite (and even small) number of bits might be sufficient to define the generating algorithm of the amplitude set. So if the information bound on the universe is interpreted as an upper limit on the *algorithmic* information (as opposed to the Shannon

information), then a measure-zero subset of initial states can be specified without violating the cosmological information bound. But this loophole leaves many unanswered questions. For example, a mathematical specification is one thing, a physical process to implement that specification – and to do so in an acceptable period of time – is another. To take the cited example, it is far from clear that there exists *any* physical process that can create an entangled quantum state in which the amplitudes (enumerated in some sequence) are the digits of π. And even if this further problem is satisfactorily addressed, one has to confront the fact that as the initial state evolves, and the amplitudes change, so the set of amplitudes may not remain algorithmically compressible. To be sure, a unitary evolution of an initially algorithmically compressible state will, by definition, preserve algorithmic compressibility (because the unitary operation is an algorithm). But such a pure system is unstable: the inevitability of random errors due to the fact that the quantum system is not closed will raise the algorithmic complexity, and seemingly raise it above the bound (1) in pretty short order.[28] This uncovers a deeper set of issues, which is whether a quantum state that cannot be specified, and is in principle unknowable, and the amplitude set of which exceeds the total information capacity of the universe, may nevertheless still be said to exist and conform to physical law. According to the Landauer point of view I am articulating here, the answer is no.

4.5. Unfinished business

I have been asked what, exactly, would go wrong if one tried to build and operate a quantum computer with, say, 500 entangled qubits. First let me make a general point. In science, one always has to distinguish between mathematical possibility contained in a theory, and physical possibility. For example, general relativity contains mathematical models with closed timelike world lines, but these may be inconsistent with cosmological boundary conditions or some other global requirement.[29] So the fact that a unitary transformation that implements a desirable quantum computation may exist mathematically does not necessarily mean it can be implemented physically, even in principle. And in fact, a prima face example would seem to be the expectation that the resources needed to prepare an initial quantum state are expected to grow with its complexity, and would require more and more of the surrounding universe to be commandeered, and more yet for the error correction of its evolution. Inevitably, the gravitational effects

of the commandeered matter will eventually become important. Before the complexity of the state reached the cosmological bound of 10^{122}, the entire resources of the observable universe would necessarily be exhausted. Thus, almost all quantum initial states, and hence almost all unitary transformations, seem to be ruled out by the cosmological constraint (1) (if one accepts it). It is important to realize, however, that this restriction may not be an impediment to preparing an algorithmically simple state, providing a physical mechanism can be found to implement the preparation algorithm. These criteria will undoubtedly be satisfied for the (very limited) examples of known quantum algorithms, such as Shor's algorithm for factorization, which is algorithmically simple by definition, since its input state can be specified and there is a simple association between the input data and the initial quantum state. What is less clear is whether this ease of preparation of the initial state is representative of a broader class of problems of interest, or remains confined to a handful of special cases.

A more radical conjecture about what might go wrong concerns the subsequent evolution of the state, which entails an escalation of the algorithmic complexity through the cosmological information bound due to random errors in the manner I mentioned above. Under these circumstances, it may be that the unitary evolution of the state actually breaks down (over and above the breakdown caused by tracing out the degrees of freedom associated with the errors caused by environmental disturbances). This would manifest itself in the form of an additional source of errors, ultimately of cosmological origin, in a manner such that all error-correcting protocols applied to these errors would fail to converge. What I am suggesting here seems to be close to the concept of unavoidable intrinsic decoherence proposed by Milburn.[30] Some clarification of these issues may emerge from the further study of the recent discovery that the entropy of quantum entanglement of a harmonic lattice also scales like area rather than volume[31], which would seem to offer support for the application of the holographic principle to entangled states. It would be good to know how general the entanglement-area relationship might be. Finally, I should point out that the information bound (1) was derived using quantum field theory, but that same bound applies to quantum field theory. Ideally one should derive the bound using a self-consistent treatment. If one adopts the philosophy that information is primary and ontological, then such a self-consistency argument should be incorporated in a larger program directed at unifying mathematics and physics. If, following Landauer, one accepts that mathematics is meaningful only if it is

the product of real computational processes (rather than existing independently in a Platonic realm) then there is a self-consistent loop: the laws of physics determine what can be computed, which in turn determines the informational basis of those same laws of physics. Benioff has considered a scheme in which mathematics and the laws of physics co-emerge from a deeper principle of mutual self-consistency,[32] thus addressing Wigner's question of why mathematics is so "unreasonably effective" in describing the physical world.[33] I have discussed these deeper matters elsewhere.[34]

Acknowledgments

I should like to thank Scott Aaronson, Ted Jacobson, Gerard Milburn, William Phillips, Sandu Popescu and Leonard Susskind for helpful comments, conversations and guidance.

Footnotes

(1) Wittgenstein, L. (1921) *Tractatus Logico-Philosophicus,* English translation: David Pears and Brian McGuinness (Routledge, London 1961).
(2) Aaronson, S. (2005) 'Are quantum states exponentially long vectors?' *Proceedings of the Oberwolfach Meeting on Complexity Theory* (to be published).
(3) Chaitin, G. (2005) *Meta Math! The Quest for Omega* (Pantheon Books, New York), 115.
(4) *Cosmic Jackpot* by Paul Davies (Houghton Mifflin, New York 2007).
(5) Laplace, P. (1825) *Philosophical Essays on Probabilities* (trans. F.L. Emory and F.W. Truscott, Dover, New York 1985).
(6) I am grateful to Michael Berry for drawing my attention to this example.
(7) 'The limits of reason,' by Gregory Chaitin, *Scientific American* March 2006, p. 74.
(8) Lloyd, S. (2002) 'Computational capacity of the universe', *Phys. Rev. Lett.* **88**, 237901;
Lloyd, S. (2006) *The Computational Universe* (Random House, New York).
(9) Bekenstein, J. (1973) *Phys. Rev. D* **8**, 2333.
(10) Hawking, S.W. (1975) *Comm. Math. Phys.* **43**, 199.
(11) Bekenstein, J (1981) *Phys. Rev. D* **23**, 287.
(12) Gibbons, G.W, and Hawking, S.W. (1977) *Phys. Rev. D* **15**, 2738., Bousso, R. (1999) *J. High Energy Phys.* **7**, 4., Davies, P.C.W. and

Davis, T.M. (2003) *Found. Phys.* **32**, 1877 (2003).
(13) Bousso, R. (1999) *J. High Energy Phys.* **7**, 4.
(14) Davies, P.C.W. and Davis, T.M. (2003) *Found. Phys.* **32**, 1877 (2003).
(15) 't Hooft, G. (1993) 'Dimensional reduction in quantum gravity,' gr-qc 9310026.
(16) Susskind, L. (1995) *J. Math. Phys.* (NY) **36**, 6377.
(17) Wheeler, J.A. (1994) *At Home in the Universe* (AIP Press, New York), 295.
(18) Fredkin, E. (1990) *Physica D* **45**, 254.
(19) Wolfram, S. (2002) *A New Kind of Science* (Wolfram Media Inc., Champaign, Ill.).
(20) Landauer, R. (1967) *IEEE Spectrum* **4**, 105.
(21) Benioff, P. (2003) 'Resource limited theories and their extensions,' quant-ph 0303086.
(22) Jacobson, T. (1991) 'Black hole evaporation and ultra-short distances,' *Phys. Rev. D* **44**, 1731.
(23) See, for example, Birrell, N.D. and Davies, P.C.W. (1982) *Quantum Fields in Curved Space* (Cambridge University Press, Cambridge).
(24) Padmanabham, T. (2004) 'Vacuum fluctuations of energy density can lead to the observed cosmological constant,' hep-th 0406060.
(25) Guberina, B., Horvat, R. and Nicolié, H. (2006) 'Dynamical dark energy with a constant vacuum energy density,' astro-ph 0601598.; Hsu, S.D. (2004) *Phys. Lett. B* **594**, 13.; Li, M. (2004) *Phys. Lett. B* **603**, 1.; Thomas, S. (2002) *Phys. Rev. Lett.* **89**, 081301-1.
(26) A review of the field may be found in Nielsen, M.A. and Chuang, I.L. (2000) *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge).
(27) Linden, N. and Popescu, S. (1998) *Fortschr. Phys.* **46**, 567.
(28) Dyakonov, M.I. (2006) 'Is fault-tolerant quantum computation really possible?' quant-ph/0610117.
(29) Davies, Paul (2001) *How to Build a Time Machine* (Penguin, London).
(30) Milburn, G.J. (1991) *Phys. Rev. A* **44**, 5401; Milburn, G.J. (2006) *New J. Phys.* **8**, 96.
(31) Cramer, M. and Eisert, J. (2006) 'Correlations, spectral gap and entanglement in harmonic quantum systems on generic lattices,' quant-ph 0509167.
(32) Benioff, P. (2002) *Found. Phys.* **32**, 989.
(33) Wigner, E (1960) 'The Unreasonable Effectiveness of Mathematics in the Natural Sciences,' in *Communications in Pure and Applied Mathe-*

matics, **13**, No. 1.

(34) Davies, Paul (2006) *The Goldilocks Enigma: Why is the universe just right for life?* (Allen Lane, London). US edition: *Cosmic Jackpot* (Houghton Mifflin, New York, 2007).

Chapter 5

What is a Computation?

Martin Davis[1]

Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, USA; martin@eipye.com

On numerous occasions during the Second World War, members of the German high command had reason to believe that the allies knew the contents of some of their most secret communications. Naturally, the Nazi leadership was most eager to locate and eliminate this dangerous leak. They were convinced that the problem was one of treachery. The one thing they did not suspect was the simple truth: the British were able to systematically decipher their secret codes. These codes were based on a special machine, the "Enigma," which the German experts were convinced produced coded messages that were entirely secure. In fact, a young English mathematician, Alan Turing, had designed a special machine for the purpose of decoding messages enciphered using the Enigma. This is not the appropriate place to speculate on the extent to which the course of history might have been different without Turing's ingenious device, but it can hardly be doubted that it played an extremely important role.

In this essay we will discuss some work which Alan Turing did a few years before the Second World War whose consequences are still being developed. What Turing did around 1936 was to give a cogent and complete logical analysis of the notion of "computation." Thus it was that although people have been computing for centuries, it has only been since 1936 that we have possessed a satisfactory answer to the question: "What is a computation?"

[1]Originally published in L. A. Steen, *Mathematics Today: Twelve Informal Essays,* Springer-Verlag, New York, 1978, pp. 241–267. Re-published with the kind permission of *Springer*.

Alan M. Turing

Alan M. Turing was born in 1912, the second son in an upper class English family. After a precocious childhood, he had a distinguished career as a student at Cambridge University. It was shortly after graduation that Turing published his revolutionary work on computability. Turing's involvement in the deciphering of German secret codes during the Second World War has only recently become public knowledge. His work has included important contributions to mathematical logic and other branches of mathematics. He was one of the first to write about the possibility of computer intelligence and his writings on the subject are still regarded as fundamental. His death of cyanide poisoning in June 1954 was officially adjudged suicide.

Turing's analysis provided the framework for important mathematical investigations in a number of directions, and we shall survey a few of them.

Turing's analysis of the computation process led to the conclusion that it should be possible to construct "universal" computers which could be programmed to carry out any possible computation. The existence of a logical analysis of the computation process also made it possible to show that certain mathematical problems are incapable of computational solution, that they are, as one says, *unsolvable.* Turing himself gave some simple examples of unsolvable problems. Later investigators found that many mathematical problems for which computational solutions had been sought unsuccessfully for many years were, in fact, unsolvable. Turing's logical proof of the existence of "universal" computers was prophetic of the modern all-purpose digital computer and played a key role in the thinking of such pioneers in the development of modern computers as John von Neumann. (Likely these ideas also played a role in Turing's seeing how to translate his cryptographic work on the German codes into a working machine.) Along with the development of modern computers has come a new branch of applied mathematics: *theory of computation*, the application of mathematics to the theoretical understanding of computation. Not surprisingly, Turing's analysis of computation has played a pivotal role in this development.

Although Turing's work on giving a precise explication of the notion of computation was fundamental because of the cogency and completeness of his analysis, it should be stated that various other mathematicians were independently working on this problem at about the same time, and that a number of their formulations have turned out to be logically equivalent to that of Turing. In fact the specific formulation we will use is closest to one originally due to the American mathematician Emil Post.

The Turing – Post Language

Turing based his precise definition of computation on an analysis of what a human being actually does when he computes. Such a person is following a set of rules which must be carried out in a completely mechanical manner. Ingenuity may well be involved in setting up these rules so that a computation may be carried out efficiently, but once the rules are laid down, they must be carried out in a mercilessly exact way. If we watch a human being calculating something (whether he is carrying out a long division, perform-

Emil L. Post

Emil L. Post was born in Poland in 1897, but arrived in New York City at the age of seven, and lived there for the remainder of his life. His life was plagued by tragic problems: he lost his left arm while still a child and was troubled as an adult by recurring episodes of a disabling mental illness. While still an undergraduate at City College he worked out a generalization of the differential calculus which later turned out to be of practical importance. His doctoral dissertation at Columbia University initiated the modern metamathematical method in logic. His researches while a postdoctoral fellow at Princeton in the early 1920's anticipated later work of Gödel and Turing, but remained unpublished until much later, partly because of the lack of a receptive atmosphere for such work at the time, and partly because Post never completed the definitive development he was seeking. His work on computability theory included the independent discovery of Turing's analysis of the computational process, various important unsolvability

ing an algebraic manipulation, or doing a calculus problem), we observe symbols being written, say on a piece of paper, and the behavior of the person doing the calculating changes as he notes various specific symbols appearing as results of computation steps.

The problem which Turing faced and solved was this: how can one extract from this process what is essential and eliminate what is irrelevant? Of course some things are clearly irrelevant; obviously it does not matter whether our calculator is or is not drinking coffee as he works, whether he is using pencil or pen, or whether his paper is lined, unlined, or quadruled. Turing's method was to introduce a series of restrictions on the calculator's behavior, each of which could clearly be seen to be inessential. However, when he was done all that was left were a few very simple basic steps performed over and over again many times.

We shall trace Turing's argument. In the first place, he argued that we can restrict the calculator to write on a linear medium, that is, on a tape, rather than on a two-dimensional sheet of paper. Instead of paper tape (such as is used in an adding machine) we can, if we prefer, think of magnetic tape as used in a tape recorder. (Of course, in this latter case, the symbols occur as magnetic signals rather than as marks on paper, but conceptually this makes no difference whatsoever.) It is easy to convince oneself that the use of a two-dimensional sheet of paper plays no essential role in the computational process and that we really are not giving up any computational power by restricting ourselves to a linear tape. Thus the "two-dimensional" multiplication:

$$\begin{array}{r} 26 \\ \times\ 32 \\ \hline 52 \\ 780 \\ \hline 832 \end{array}$$

can be written on a "tape" as follows:

$$26 \times 32 = 52 + 780 = 832.$$

We suppose that the linear tape is marked off into individual squares and that only one symbol can occupy a square. Again, this is a matter of convenience and involves no particular limitations. So, our multiplication example might look like this:

2	6	×	3	2	=	5	2	+	7	8	0	=	8	3	2

.

The next restriction we impose (here we are actually going a bit further than Turing did) is that the only symbols which may appear on our tape are 0 and 1. Here we are merely making use of the familiar fact that all information can be "coded" in terms of two symbols. It is this fact, for example, which furnishes the basis for Morse code in which the letters of the alphabet are represented as strings of "dots" and "dashes." Another example is binary arithmetic which forms the basis of modern digital computation.

Our next restriction has to do with the number of different symbols our calculator can take note of (or as we shall say, "scan") in a single observation. How many different symbols can a human calculator actually take in at one time? Certainly no one will be able to take in at a glance the distinction between two very long strings of zeros and ones which differ only at one place somewhere in the middle. One can take in at a glance, perhaps, five, six, seven, or eight symbols. Turing's restriction was more drastic. He assumed that in fact one can take in only a single symbol at a glance. To see that this places no essential restriction on what our calculator can accomplish, it suffices to realize that whatever he does as a result of scanning a group of, say, five symbols can always be broken up into separate operations performed viewing the symbols one at a time.

What kinds of things can the calculator actually do? He can replace a 0 by a 1 or a 1 by a 0 on the square he is scanning at any particular moment, or he can decide to shift his attention to another square. Turing assumed that this shifting of attention is restricted to a square which is the immediate neighbor, either on the left or on the right, of the square previously scanned. Again, this is obviously no essential restriction: if one wants to shift one's attention to a square three to the right, one simply shifts one to the right three successive times. Also the calculator may observe the symbol in the square being scanned and make a decision accordingly. And presumably this decision should take the form: "Which instruction shall I carry out next?" Finally, the calculator may halt, signifying the end of the computation.

To summarize: any computation can be thought of as being carried out by a human calculator, working with strings of zeros and ones written on a linear tape, who executes instructions of the form:

- Write the symbol 1
- Write the symbol 0

- Move one square to the right
- Move one square to the left
- Observe the symbol currently scanned and choose the next step accordingly
- Stop

The procedure which our calculator is carrying out then takes the form of a list of instructions of these kinds. As in modern computing practice, it is convenient to think of these kinds of instructions as constituting a special *programming language.* A list of such instructions written in this language is then called a *program.*

We are now ready to introduce the Turing–Post Programming Language. In this language there are seven kinds of instructions:

PRINT 1
PRINT 0
GO RIGHT
GO LEFT
GO TO STEP i IF 1 IS SCANNED
GO TO STEP i IF 0 IS SCANNED
STOP

A Turing–Post program is then a list of instructions, each of which is of one of these seven kinds. Of course in an actual program the letter i in a step of either the fifth or sixth kind must be replaced by a definite (positive whole) number.

In order that a particular Turing–Post program begin to calculate, it must have some "input" data. That is, the program must begin scanning at a specific square of a tape already containing a sequence of zeros and ones. The symbol 0 functions as a "blank"; although the entire tape is infinite, there are never more than a finite number of ones that appear on it in the course of a computation. (A reader who is disturbed by the notion of an infinite tape can replace it for our purposes by a finite tape to which blank squares—that is, squares filled with zeros—are attached to the left or the right whenever necessary.)

Figure 1 exhibits a Turing–Post program consisting of ten instructions which we will use repeatedly for illustrative purposes. The presence of the "**GO TO**" instruction makes it possible for the same instruction to be executed over and over again in the course of a single computation. This can be seen in some detail in Figure 2 which shows the successive steps in one

1.	**PRINT 0**
2.	**GO LEFT**
3.	**GO TO STEP 2 IF 1 IS SCANNED**
4.	**PRINT 1**
5.	**GO RIGHT**
6.	**GO TO STEP 5 IF 1 IS SCANNED**
7.	**PRINT 1**
8.	**GO RIGHT**
9.	**GO TO STEP 1 IF 1 IS SCANNED**
10.	**STOP**

Figure 1. Doubling Program. The underlying idea is to double the number of ones by simply copying them one at a time. Each 1 to be copied is (temporarily) replaced by a 0 which acts as a place marker (Step 1). Next the computation moves left over all the ones (which as the computation progresses will include newly printed ones) seeking the first unused (i.e., blank) square (Steps 2, 3—which will be repeated over and over again until a blank square is encountered). The 1 is now copied (Step 4). Next the computation returns rightward until it encounters the 0 which takes the place of the 1 which has just been copied (Steps 5, 6—which again are repeated). The copied 1 is restored (Step 7). The computation moves one square to the right seeking another 1 to copy (Step 8). If there is another 1 to be copied the computation goes back to Step 1; otherwise it advances to Step 10 and halts (Steps 9, 10).

particular computation by the program of Figure 1. The computation is completely determined by the initial arrangement of symbols on the tape together with a specification of which square is initially scanned. In Figure 2 this latter information is given by an upward arrow (↑) below the scanned square. (Of course only a finite number of symbols from the tape can actually be explicitly exhibited; in Figure 2, we exhibit six adjacent symbols, and assume that all squares not explicitly shown are blank, that is contain the symbol 0.) Such combined information, consisting of the symbols on the tape (pictorially represented by showing a finite number of consecutive squares, the remainder of which are presumed to be blank) and the identity of the scanned square (designated by an arrow just below it) is called a *tape configuration.*

Figure 2 gives a list of such tape configurations, with the initial configu-

ration at the top, each of which is transformed by an appropriate step of the program (from Figure 1) into the configuration shown below it. The program steps are listed alongside the tape configurations. The computation begins by executing the first step (which in our case consists in replacing the 1 on the scanned square by 0) and continues through the successive steps of the program, except as "**GO TO**" instructions cause the computation to return to earlier instructions. Ultimately, Step 9 is executed with the tape configuration as shown at the bottom of Figure 2. Since 0 is being scanned, the computation continues to Step 10 and then halts.

The computation shown in Figure 2 begins with two ones on the tape and ends with four. It is because this happens generally that we call the program in Figure 1 a "doubling program." To put it precisely: beginning with a tape configuration the nonblank portion of which consists of a row of ones with the scanned square containing the leftmost of the ones, the doubling program will eventually halt with a block of twice as many ones on the tape as were there to begin with. It is by no means obvious at a glance (even to an experienced computer programmer) that our doubling program really behaves in the manner just stated.

The fact that this doubling program is so short and accomplishes such a simple task should not be permitted to obscure the point of Turing's analysis of the computation process: we have reason to be confident that any computation whatsoever can be carried out by a suitable Turing–Post program.

As we have seen, once a **STOP** instruction is executed, computation comes to a halt. If, however, no **STOP** instruction is ever encountered in the course of a computation, the computation will (in principle, of course) continue forever. The question "When can we say that a computation will eventually halt?" will play a crucial role later in our discussion. To see how this can be answered in a simple example, consider the following three-step Turing–Post program:

1. GO RIGHT
2. GO TO STEP 1 IF 0 IS SCANNED
3. STOP

This program will halt as soon as motion to the right reaches a square containing the symbol 1. For once that happens the program will move on to Step 3 and halt. That being the case, suppose we begin with a tape on which there are no ones to the right of the initially scanned square. (For

example, the entire tape could be blank or there could be some ones but all to the left of the initially scanned square.) In this case, the first two steps will be carried out over and over again forever, since a 1 will never be encountered. After step 2 is performed, step 1 will be performed again. This makes it clear that a computation from a Turing–Post program *need not actually ever halt.* In the case of this simple three-step program it is very easy to tell from the initial tape configuration whether the computation will eventually halt or continue forever: to repeat, if there is a 1 to the right of the initially scanned square the computation will eventually halt; whereas if there are only blanks to the right of the initially scanned square the computation will continue forever. We shall see later that the question of predicting whether a particular Turing–Post program will eventually halt contains surprising subtleties.

Codes for Turing – Post Programs

All of the dramatic consequences of Turing's analysis of the computation process proceed from Turing's realization that it is possible to encode a Turing–Post program by a string of zeros and ones. Since such a string can itself be placed on the tape being used by another (or even the same) Turing–Post program, this leads to the possibility of thinking of Turing–Post programs as being capable of performing computations on other Turing–Post programs.

There are many ways by which Turing–Post programs can be encoded by strings of zeros and ones. We shall describe one such way. We first represent each Turing–Post instruction by an appropriate sequence of zeros and ones according to the following code:

Code	Instruction
000	**PRINT 0**
001	**PRINT 1**
010	**GO LEFT**
011	**GO RIGHT**
$101\underbrace{0\ldots0}_{i}1$	**GO TO STEP i IF 0 IS SCANNED**
$110\underbrace{1\ldots1}_{i}0$	**GO TO STEP i IF 1 IS SCANNED**
100	**STOP**

This table gives the representation of each Turing–Post instruction by a string of zeros and ones. For example the code for the instruction

GO TO STEP 3 IF 0 IS SCANNED

is: 1010001. To represent an entire program, we simply write down in order the representation of each individual instruction and then place an additional 1 at the very beginning and 111 at the very end as punctuation marks.

For example, here is the code for the doubling program shown in Figure 1:

100001011011000101111011111000101111010100111

To make this clear, here is the breakdown of this code:

Begin	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8
1	000	010	110110	001	011	110111110	001	011

Step 9	Step 10	End
11010	100	111

It is important to notice that the code of a Turing–Post program can be deciphered in a unique, direct, and straightforward way, yielding the program of which it is the code. First remove the initial 1 and the final 111 which are just punctuation marks. Then, proceeding from left to right, mark off the first group of 3 digits. If this group of 3 digits is 000, 001, 010, 011, or 100 the corresponding instruction is: **PRINT 0**, **PRINT 1**, **GO LEFT**, **GO RIGHT**, or **STOP**, respectively. Otherwise the group of 3 digits is 101 or 110, and the first instruction is a "**GO TO**." The code will then have one of the forms:

$$101\underbrace{0\ldots0}_{i}1 \qquad 110\underbrace{1\ldots1}_{i}0$$

corresponding to

GO TO STEP i IF 0 IS SCANNED

and

GO TO STEP i IF 1 IS SCANNED

respectively. Having obtained the first instruction, cross out its code and continue the process, still proceeding from left to right. Readers who wish to test their understanding of this process may try to decode the string:

$$10100011010011000010101010111$$

The Universal Program

We are now ready to see how Turing's analysis of the computation process together with the method for coding Turing–Post programs we have just introduced leads to a conclusion that at first sight seems quite astonishing. Namely, there exists a single (appropriately constructed) Turing–Post program which can compute anything whatever that is computable. Such a program U (for "universal") can be induced to simulate the behavior of any given Turing–Post program P by simply placing $code(P)$, the string of zeros and ones which represents P, on a tape and permitting U to operate on it. More precisely, the non-blank portion of the tape is to consist of $code(P)$ followed by an input string v on which P can work. (For clarity, we employ capital letters to stand for particular Turing–Post programs and lowercase letters to stand for strings of zeros and ones.) For example, the string

$$\underbrace{1}_{\text{Begin}}\ \underbrace{0000101101100010111101111100010111101010011}_{\text{Coded instructions of doubling program}}\ \underbrace{111}_{\text{End}}\ \underbrace{11}_{\text{Input}}$$

signifies that U should simulate the behavior of the doubling program when 11 is the input. Thus, at the end of the computation by U, the tape should look just like the final tape in Figure 2.

Now, a universal Turing–Post program U is supposed to perform in this way not only for our doubling program, but for *every* Turing–Post program. Let us be precise: U is to begin its computation presented with a tape whose nonblank portion consists of $code(P)$ for some Turing–Post program P (initially scanning the first symbol, necessarily 1, of this code) followed by a string v. U is then supposed to compute exactly the same result as the program P would get when starting with the string v as the nonblank part of the tape (scanning the initial symbol of v). Such a program U can then be used to simulate any desired Turing–Post program P by simply placing the string $code(P)$ on the tape.

What reason do we have for believing that there is such a program U? To help convince ourselves, let us begin by thinking how a human calculator could do what U is supposed to do. Faced with the tape contents on which U is supposed to work, such a person could begin by scanning this string of zeros and ones, from left to right, searching for the first place

that 3 consecutive ones appear. This triple 111 marks the end of $code(P)$ and the beginning of the input string. Our human calculator can then write $code(P)$ on one sheet of paper and the input string on another. As already explained, he can decode the string $code(P)$ and obtain the actual Turing–Post program P. Finally, he can "play machine," carrying out the instructions of P, applied to the given input string in a robotlike fashion. If and when the computation comes to a halt, our calculator can report the final tape contents as output. This shows that a human calculator can do what we would like U to do. But now, invoking Turing's analysis of the computation process, we are led to believe that there must be a Turing–Post program which can carry out the process we have just described, a universal Turing–Post program.

The evidence we have given for the existence of such a program is rather unsatisfactory because it depends on Turing's analysis of the computation process. It certainly is not a mathematical proof. But in fact, if one is willing to do some tedious but not very difficult work, one can circumvent the need to refer to Turing's analysis at all and can, in fact, write out in detail an explicit universal Turing–Post program. This was done in fact by Turing himself (in a slightly different, but entirely equivalent context) in his fundamental 1936 paper. And subsequently, it has been redone many times. The success of the construction of the universal program is in itself evidence for the correctness of Turing's analysis. It is not appropriate here to carry out the construction of a universal program in detail; we hope, merely, that the reader is convinced that such a program exists. (Experienced computer programmers will have no difficulty in writing their own universal program if they wish to do so.)

We have conceived of Turing–Post programs as consisting of lists of written instructions. But clearly, given any particular Turing–Post program P, it would be possible to build a machine that would actually carry out the instructions of P in sequence. In particular, this can be done for our universal program U. The machine we get in this way would be an example of an all-purpose or universal computing machine. The code for a particular program P placed on its tape could then be thought of as a "program" for doing the computation which P does. Thus, Turing's analysis leads us, in a very straightforward manner, to the concept of an all-purpose computer which can be programmed to carry out any computation whatever.

The Halting Problem

We are now in a position to demonstrate a truly astonishing result: we are able to state a simple problem, the so-called *halting problem*, for which we can prove that no computational solution exists.

The halting problem for a particular Turing–Post program is the problem of distinguishing between initial tape configurations which lead to the program's eventually halting and initial tape configurations which lead the program to compute forever. We saw above that certain input strings may cause a particular program to run forever, due to an infinite loop caused by the "**GO TO**" instruction. It would surely be desirable to have a method for determining in advance which input data leads the program to halt and which does not. This is the halting problem: given a particular Turing–Post program, can we computationally test a given tape configuration to see whether or not the program will eventually halt when begun with that tape configuration.

The answer is no. *There is no computation procedure for testing a given tape expression to determine whether or not the universal program U will eventually halt when begun with that tape configuration.* The fact that there is no such procedure for the universal program shows of course that there can't be such procedures for Turing–Post programs in general, since the universal program is itself a Turing–Post program. Before we see how this unsolvability theorem can be proved, it is worthwhile to reflect on how exciting and remarkable it is that it should be possible to prove such a result. Here is a problem which is easy to state and easy to understand which we *know* cannot be solved. Note that we are not saying simply that we don't know how to solve the problem, or that the solution is difficult. We are saying: *there is no solution.*

Readers may be reminded of the fact that the classical problems of angle trisection and circle squaring also turned out to have no solution. This is a good analogy, but with a very significant difference: the impossibility proofs for angle trisection and circle squaring are for constructions using specific instruments (straightedge and compass); using more powerful instruments, there is no difficulty with either of these geometric construction problems. Matters are quite different with the halting problem; here what we will show is that there is no solution using any methods available to human beings.

The proof of the unsolvability of the halting problem is remarkably simple. It uses the method known as indirect proof or *reductio ad absurdum.* That is, we suppose that what is stated in italics above is false, that in fact, we possess a computing procedure which, given an initial tape configuration will enable us to determine whether or not the universal program will eventually halt when started in that configuration. Then we show that this supposition is impossible; this is done in the box on p. 114.

Other Unsolvable Problems

In the 1920's the great German mathematician David Hilbert pointed to a certain problem as the fundamental problem of the newly developed field of mathematical logic. This problem, which we may call the decision problem for elementary logic, can be explained as follows: a finite list of statements called *premises* is given together with an additional statement called the *conclusion.* The logical structure of the statements is to be explicitly exhibited in terms of "not," "and," "or," "implies," "for all," and "there exists." Hilbert wanted a computing procedure for testing whether or not the conclusion can be deduced using the rules of logic from the premises. Hilbert regarded this problem as especially important because he expected that its solution would lead to a purely mechanical technique for settling the truth or falsity of the most diverse mathematical statements. (Such statements could be taken as the conclusion, and an appropriate list of axioms as the premises to which the supposed computing procedure could be applied.) Thus the very existence of an unsolvable mathematical problem (in particular, the halting problem) immediately suggested that Hilbert's decision problem for elementary logic was itself unsolvable. This conclusion turned out to be correct, as was carefully shown by Turing and, quite independently, by the American logician Alonzo Church. Turing represented the theory of Turing–Post programs in logical terms and showed that a solution to the decision problem for elementary logic would lead to a solution of the halting problem. (This connection between logic and programs was rediscovered many years later and now forms the basis for certain investigations into the problem of proving the correctness of computer programs.)

The unsolvability of the decision problem for elementary logic was important, not only because of the particular importance of this problem, but also because (unlike the halting problem) it was an unsolvable problem that people had actually tried to solve. A decade went by before another such

example turned up. Early in the century the Norwegian Axel Thue had emphasized the importance of what are now called "word problems." In 1947, Emil Post showed how the unsolvability of the halting problem leads to the existence of an unsolvable word problem. Post's proof is discussed in the box on p. 116. Here we merely explain what a word problem is.

In formulating a word problem one begins with a (finite) collection, called an *alphabet*, of symbols, called *letters.* Any string of letters is called a *word* on the alphabet. A word problem is specified by simply writing down a (finite) list of equations between words. Figure 3 exhibits a word problem specified by a list of 3 equations on the alphabet a, b, c. From the given equations many other equations may be derived by making substitutions in any word of equivalent expressions found in the list of equations. In the example of Figure 3, we derive the equation $bac = abcc$ by replacing the part ba by abc as permitted by the first given equation.

We have explained how to *specify* the data for a word problem, but we have not yet stated what the problem is. It is simply the problem of determining for two arbitrary given words on the given alphabet, whether one can be transformed into the other by a sequence of substitutions that are legitimate using the given equations. We show in the box on p. 116 that we can specify a particular word problem that is unsolvable. In other words, no computational process exists for determining whether or not two words can be transformed into one another using the given equations. Work on unsolvable word problems has turned out to be extremely important, leading to unsolvability results in different parts of mathematics (for example, in group theory and in topology).

Another important problem that eventually turned out to be unsolvable first appeared as the tenth in a famous list of problems given by David Hilbert in 1900. This problem involves so-called "Diophantine" equations. An equation is called Diophantine when we are only interested in solutions in integers (i.e., whole numbers). It is easy to see that the equation

$$4x - 2y = 3$$

has no solutions in integers (because the left side would have to be even while the right side is odd). On the other hand the equation

$$4x - y = 3$$

has many (even infinitely many) solutions in integers (e.g., $x = 1$, $y = 1$;

$x = 2$, $y = 5$). The Pythagorean equation

$$x^2 + y^2 = z^2$$

also has infinitely many integer solutions (of which $x = 3$, $y = 4$, $z = 5$ was already known to the ancient Egyptians). Hilbert's tenth problem was to find a computing procedure for testing a Diophantine equation (in any number of unknowns and of any degree) to determine whether or not it has an integer solution.

Since I have been directly involved with this problem and related matters over the past thirty years, my discussion of Hilbert's tenth problem will necessarily have a rather personal character. I first became interested in the problem while I was an undergraduate at City College of New York on reading my teacher Emil Post's remark in one of his papers that the problem "begs for an unsolvability proof." In my doctoral dissertation at Princeton, I proved the unsolvability of a more difficult (and hence easier to prove unsolvable) related problem. At the International Congress of Mathematicians in 1950, I was delighted to learn that Julia Robinson, a young mathematician from California, had been working on the same problem from a different direction: she had been developing ingenious techniques for expressing various complicated mathematical relationships using Diophantine equations. A decade later Hilary Putnam (a philosopher with training in mathematical logic) and I, working together, saw how we could make further progress by combining Julia Robinson's methods with mine. Julia Robinson improved our results still further, and we three then published a joint paper in which we proved that if there were even one Diophantine equation whose solutions satisfy a special condition (involving the relative size of the numbers constituting such a solution), then Hilbert's tenth problem would be unsolvable.

In subsequent years, much of my effort was devoted to seeking such a Diophantine equation (working alone and also with Hilary Putnam), but with no success. Finally such an equation was found in 1970 by the then 22-year old Russian mathematician Yuri Matiyasevich. Matiyasevich's brilliant proof that his equation satisfied the required condition involved surprisingly elementary mathematics. His work not only showed that Hilbert's tenth problem is unsolvable, but has also led to much new and interesting work.

Undecidable Statements

The work of Bertrand Russell and Alfred North Whitehead in their three-volume magnum opus *Principia Mathematica*, completed by 1920, made it clear that all *existing* mathematical proofs could be translated into the specific logical system they had provided. It was assumed without question by most mathematicians that this system would suffice to prove or disprove any statement of ordinary mathematics. Therefore mathematicians were shocked by the discovery in 1931 by Kurt Gödel (then a young Viennese mathematician) that there are statements about the whole numbers which can neither be proved nor disproved in the logical system of *Principia Mathematica* (or similar systems); such statements are called *undecidable.* Turing's work (which was in part inspired by Gödel's) made it possible to understand Gödel's discovery from a different, and indeed a more general, perspective.

Let us write $N(P, v)$ to mean that the Turing–Post program P will *never* halt when begun with v on its tape (as usual, scanning its leftmost symbol). So, for any particular Turing–Post program P and string v, $N(P, v)$ is a perfectly definite statement which is either true (in case P will never halt in the described situation) or false (in case P will eventually halt). When $N(P, v)$ is false, this fact can always be demonstrated by exhibiting the complete sequence of tape configurations produced by P leading to termination. However, when $N(P, v)$ is true no finite sequence of tape configurations will suffice to demonstrate the fact. Of course we may still be able to prove that a particular $N(P, v)$ is true by a logical analysis of P's behavior.

Let us try to be very rigorous about this notion of *proof.* Suppose that certain strings of symbols (possibly paragraphs of English) have been singled out as proofs of particular statements of the form $N(P, v)$. Suppose furthermore that we possess a computing procedure that can test an alleged proof Π that $N(P, v)$ is true and determine whether Π is or is not actually such a proof. Whatever our rules of proof may be, this requirement is surely needed for communication purposes. It must be possible in principle to perform such a test in order that Π should serve its purpose of eliminating doubts concerning the truth of $N(P, v)$. (In practice, published mathematical proofs are in highly condensed form and do not meet this strict requirement. Disputes are resolved by putting in more detail as needed. But it is essential that *in principle* it is always possible to include

sufficient detail so that proofs are susceptible to mechanical verification.)

There are two basic requirements which it is natural to demand of our supposed rules of proof:

- *Soundness:* If there is a proof Π that $N(P, v)$ is true, then P will in fact never halt when begun with v on its tape.
- *Completeness:* If P will never halt when begun with v on its tape, then there is a proof Π that $N(P, v)$ is true.

Gödel's theorem asserts that no rules of proof can be both sound and complete! In other words, if a given set of rules of proof is sound, then there will be some true statement $N(P, v)$ which has no proof Π according to the given rules of proof. (Such a true unprovable statement may be called *undecidable* since it will surely not be disprovable.)

To convince ourselves of the truth of Gödel's theorem, suppose we had found rules of proof which were both sound and complete. Suppose "proofs" according to these rules were particular strings of symbols on some specific finite alphabet. We begin by specifying a particular infinite sequence $\Pi_1, \Pi_2, \Pi_3, \ldots$ which includes all finite strings on this alphabet. Namely, let all strings of a given length be put in "alphabetical" order, and let shorter strings always precede longer ones. The sequence $\Pi_1, \Pi_2, \Pi_3, \ldots$ includes all possible proofs, as well as a lot of other things; in particular, it contains a high percentage of total nonsense—strings of symbols combined in completely meaningless ways. But, hidden among the nonsense, are all possible proofs.

Now we show how we can use our supposed rules of proof to solve the halting problem for some Turing–Post program P. We wish to find out whether or not P will eventually halt when begun on v. We have some friend begin to carry out the instructions of P on input v with the understanding that we will be informed at once if the process halts. Meanwhile we occupy ourselves by generating the sequence $\Pi_1, \Pi_2, \Pi_3, \ldots$ of possible proofs. As each Π_i is generated we use our computing procedure to determine whether or not Π_i is a proof of $N(P, v)$. Now, if P will eventually halt, our friend will discover the fact and will so inform us. And, if P will never halt, since our rules of proof are assumed to be complete, there will be a proof Π_i of $N(P, v)$ which we will discover. Having obtained this Π_i we will be sure (because the rules are sound) that P will indeed never halt. Thus, we have described a computing procedure (carried out with a little help from a friend) which would solve the halting problem for P. Since, as we

well know, P can have an unsolvable halting problem (e.g., P could be the universal program U), we have arrived at a contradiction; this completes the proof of Gödel's theorem.

Of course, Gödel's theorem does not tell us that there is any particular pair P, v for which we will never be able to convince ourselves that $N(P, v)$ is true. It is simply that, for any given sound rules of proof, there will be a pair P, v for which $N(P, v)$ is true, but not provable *using the given rules.* There may well be other sound rules which decide this "undecidable" statement. But these other rules will in turn have their own undecidabilities.

Complexity and Randomness

A computation is generally carried out in order to obtain a desired answer. In our discussion so far, we have pretty much ignored the "answer," contenting ourselves with discussing only the gross distinction between a computation which does at least halt eventually and one which goes on forever. Now we consider the question: how complex need a Turing–Post program be to produce some given output? This straightforward question will lead us to a mathematical theory of randomness and then to a dramatic extension of Gödel's work on undecidability.

We will only consider the case where there are at least 2 ones on the tape when the computation halts. The output is then to be read as consisting of the string of zeros and ones between the leftmost and rightmost ones on the tape, and not counting these extreme ones. Some such convention is necessary because of the infinite string of zeros and ones on the tape. In effect the first and last one serve merely as punctuation marks.

To make matters definite suppose that we wish to obtain as output a string consisting of 1022 ones. When we include the additional ones needed for punctuation, we see that what is required is a computation which on termination leaves a tape consisting of a block of 1024 ones and otherwise blank. One way to do this is simply to write the 1024 ones on the tape initially and do no computing at all. But surely we can do better. We can get a slight improvement by using our faithful doubling program (Figure 1). We need only write 512 ones on the tape and set the doubling program to work. We have already written out the code for the doubling program; it took 39 bits. (A *bit* is simply a zero or a one; the word abbreviates *bi*nary digi*t*.) So we have a description of a string of 1022 ones which uses $39 + 512 = 551$ bits. But surely we can do better. $1024 = 2^{10}$, so we should

be able to get 1024 ones by starting with 1 and applying the doubling program 10 times. In Figure 4 we give a 22-step program, the first nine steps of which are identical to the first nine steps of the doubling program, which accomplishes this. Beginning with a tape configuration

$$\begin{array}{l} 10\underbrace{11\ldots1}_{n} \\ \uparrow \end{array}$$

this program will halt with a block of 2^{n+1} ones on the tape.

It is not really important that the reader understand how this program works, but here is a rough account: the program works with two blocks of ones separated by a zero. The effect of Steps 1 through 9 (which is just the doubling program) is to double the number of ones to the left of the 0. Steps 10 through 21 then erase 1 of the ones to the right of the zero and return to Step 1. When all of the ones to the right of the zero have been erased, this will result in a zero being scanned at Step 11 resulting in a transfer to Step 22 and a halt. Thus the number of ones originally to the left of the zero is doubled as many times as there are ones originally to the right of the zero.

The full code for the program of Figure 4 contains 155 bits. To obtain the desired block of 1024 ones we need the input 10111111111. We are thus down to $155 + 11 = 166$ bits, a substantial improvement over 551 bits.

We are now ready for a definition: Let w be any string of bits. Then we say that w has *complexity* n (or equivalently, *information content* n) and write $I(w) = n$ if:

(1) There is a program P and string v such that the length of $code(P)$ plus the length of v is n, and P when begun with v will eventually halt with output w (that is with $1w1$) occupying the nonblank part of the tape, and
(2) There is no number smaller than n for which this is the case.

If w is the string of 1022 ones, then we have shown that $I(w) \leq 166$. In general, if w is a string of bits of length n, then we can easily show that $I(w) \leq n + 9$. Specifically, let the program P consist of the single instruction: **STOP**. Since this program does not do anything, if it begins with input $1w1$, it will terminate immediately with $1w1$ still on the tape. Since $Code(P) = 1100111$, it must be the case that $I(w)$ is less than or equal to the length of the string $11001111w1$, that is, less than or equal to $n + 9$. (Naturally, the number 9 is just a technical artifact of our particular formulation and is of no theoretical importance.)

How many strings are there of length n such that, say, $I(w) \leq n - 10$? (We assume $n > 10$; in the interesting cases n is much larger than 10.) Each such w would be associated with a program P and string v such that $Code(P)v$ is a string of bits of length less than or equal to $n - 10$. Since the total number of strings of bits of length i is 2^i, there are only:

$$2 + 4 + \ldots + 2^{n-10}$$

strings of bits of length $\leq n - 10$. This is the sum of a geometric series easily calculated to be $2^{n-9} - 2$. So we conclude: there are fewer than 2^{n-9} strings of bits w of length n such that $I(w) \leq n - 10$.

Since there are 2^n strings of bits of length n, we see that the ratio of the number of strings of length n with complexity $\leq n - 10$ to the total number of strings of length n is no greater than

$$\frac{2^{n-9}}{2^n} = \frac{1}{2^9} = \frac{1}{512} < \frac{1}{500}.$$

This is less than 0.2%. In other words, more than 99.8% of all strings of length n have complexity $> n - 10$. Now the complexity of the string of 1022 ones is, as we know, less than or equal to 166, thus much less than $1022 - 10 = 1012$. Of course, what makes this string so special is that the digit pattern is so regular that a comparatively short computational description is possible. Most strings are irregular or as we may say, *random.*

Thus we are led to an entirely different application of Turing's analysis of computation: a mathematical theory of random strings. This theory was developed around 1965 by Gregory Chaitin, who was at the time an undergraduate at City College of New York (and independently by the world famous A. N. Kolmogorov, a member of the Academy of Sciences of the U.S.S.R.). Chaitin later showed how his ideas could be used to obtain a dramatic extension of Gödel's incompleteness theorem, and it is with this reasoning of Chaitin's that we will conclude this essay.

Let us suppose that we have rules of proof for proving statements of the form $I(w) > n$ where w is a string of bits and n is a positive integer. As before, we assume that we have a computing procedure for testing an alleged proof Π to see whether it really is one. We assume that the rules of proof are sound, so that if Π is a proof of the statement $I(w) > n$, then the complexity of the string w really is greater than n. Furthermore, let us make the very reasonable assumption that we have another computing procedure which, given a proof Π of a statement $I(w) > n$, will furnish us

with the specific w and n for which $I(w) > n$ has been proved.

We now describe a new computing procedure we designate as Δ. We begin generating the sequence $\Pi_1, \Pi_2, \Pi_3, \ldots$ of possible proofs as above. For each Π_i we perform our test to determine whether or not Π_i is a proof of a statement of the form $I(w) > n$. If the answer is affirmative we use our second procedure to find the specific w and n. Finally we check to see whether $n > k_0$ where k_0 is some fixed large number. If so, we report w as our answer; otherwise we go on to the next Π_i. By Turing's analysis this entire procedure Δ can be replaced by a Turing–Post program, where the fixed number k_0 is to be chosen at least as large as the length of this program. (The fact that k_0 can be chosen as large as this is not quite obvious; the basic reason is that far fewer than k_0 bits suffice to describe the number k_0.)

Now, a little thought will convince us that this Turing–Post program can never halt: if it did halt we would have a string w for which we had a proof Π_i that $I(w) > n$ where $n > k_0$. On the other hand this very program has length less than or equal to k_0 (and hence less than n) and has computed w, so that $I(w) < n$, in contradiction to the soundness of our proof rules. Conclusion: our rules of proof can yield a proof of no statement of the form $I(w) > n$ for which $n > k_0$. This is Chaitin's form of Gödel's theorem: given a sound set of rules of proof for statements of the form $I(w) > n$, there is a number k_0 such that no such statement is provable using the given rules for any $n > k_0$.

To fully understand the devastating import of this result it is important to realize that there exist rules of proof (presumably sound) for proving statements of the form $I(w) > n$ which include all methods of proof available in ordinary mathematics. (An example is the system obtained by using the ordinary rules of elementary logic applied to a powerful system of axioms, of which the most popular is the so-called Zermelo-Fraenkel axioms for set theory.) We are forced to conclude that there is some definite number k_0, such that it is in principle impossible, by ordinary mathematical methods, to prove that any string of bits has complexity greater than k_0. This is a remarkable limitation on the power of mathematics as we know it.

Although we have discussed a considerable variety of topics, we have touched on only a tiny part of the vast amount of work which Turing's analysis of the computation process has made possible. It has become

possible to distinguish not only between solvable and unsolvable problems, but to study an entire spectrum of "degrees of unsolvability." The very notion of computation has been generalized to various infinite contexts. In the theory of formal languages, developed as part of computer science, various limitations on Turing–Post programs turn out to correspond in a natural way to different kinds of "grammars" for those languages. There has been much work on what happens to the number of steps and amount of tape needed when the programs are allowed to operate on several tapes simultaneously instead of on just one. "Nondeterministic" programs in which a given step may be followed by several alternative steps have been studied, and a great deal of work has been done attempting to show that such programs are intrinsically capable of performing much faster than ordinary Turing–Post programs. These problems have turned out to be unexpectedly difficult, and much remains to be done.

Suggestions for Further Reading

General

- Chaitin, Gregory. Randomness and mathematical proof. *Scientific American* **232** (May 1975) 47–52.
- Davis, Martin and Hersh, Reuben. Hilbert's 10 problem. *Scientific American* **229** (November 1973) 84–91.
- Knuth, Donald E. Algorithms. *Scientific American* **236** (April 1977) 63–80, 148.
- Knuth, Donald E. Mathematics and computer science: coping with finiteness. *Science* **194** (December 17, 1976) 1235–1242.
- Turing, Sara. *Alan M. Turing.* W. Heffer, Cambridge, 1959.
- Wang, Hao. Games, logic and computers. *Scientific American* **213** (November 1965) 98–106.

Technical

- Davis, Martin. *Computability and Unsolvability.* McGraw-Hill, Manchester, 1958.
- Davis, Martin. Hilbert's tenth problem is unsolvable. *American Mathematical Monthly* **80** (March 1973) 233–269.
- Davis, Martin. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions.* Raven Pr, New York, 1965.

- Davis, Martin. Unsolvable problems. In *Handbook of Mathematical Logic,* by Jon Barwise (Ed.). North-Holland, Leyden, 1977.
- Minsky, Marvin. *Computation: Finite and Infinite Machines.* Prentice-Hall, Englewood Cliffs, 1967.
- Rabin, Michael O. Complexity of computations, *Comm. Assoc. Comp. Mach.* **20** (1977) 625–633.
- Trakhtenbrot, B. A. *Algorithms and Automatic Computing Machines.* D. C. Heath, Lexington, 1963.

Unsolvability of Halting Problem

Suppose we possess a computing procedure which solves the halting problem for the universal program U. Then we can imagine more complicated procedures of which this supposed procedure is a part. Specifically, we consider the following procedure which begins with a string v of zeros and ones:

(1) Try to decode v as the code for a Post–Turing program, i.e., try to find P with $code(P) = v$. If there is no such P, go back to the beginning of Step 1; otherwise go on to Step 2.
(2) Make a copy of v and place it to the right of v getting a longer string which we can write as vv (or equivalently as $code(P)v$ since $code(P) = v$).
(3) Use our (pretended) halting problem procedure to find out whether or not the universal program U will eventually halt if it begins with this string vv as the nonblank portion of the tape, scanning the leftmost symbol. If U will eventually halt, go back to the beginning of Step 3; otherwise stop.

This proposed procedure would eventually stop if, first, $v = code(P)$ for some Turing–Post program P (so we will leave Step 1 and go on to Step 2), and, second, if also U will never halt if it begins to scan the leftmost symbol of vv. Since U beginning with $code(P)v$ simulates the behavior of P beginning with v, we conclude that our supposed procedure applied to the string v will eventually stop if and only if $v = code(P)$ where P is a computing procedure that will never stop beginning with v on its tape.

By Turing's analysis, there should be a Turing–Post program P_0 which carries out this very procedure. That is, P_0 will eventually halt beginning with the input v if and only if U will never halt beginning with the input vv. Now let $v_0 = code(P_0)$. Does U eventually halt beginning with the input v_0v_0? By what we have just said, P_0 *eventually halts beginning with the input* v_0 *if and only if* U *will never halt beginning with the input* v_0v_0. But, as we will show, this contradicts our explanation of how U works as a universal program. Since $v_0 = code(P_0)$, U will act, given the input v_0v_0, to simulate the behavior of P_0 when begun on input v_0. So U *will eventually halt beginning with the input* v_0v_0 *if and only if* P_0 *will eventually halt beginning with the input* v_0. But this contradicts the previous italicized

statement. The only way out of this contradiction is to conclude that what we were pretending is untenable. In other words, the halting problem for U is not solvable.

An Unsolvable Word Problem

One way to find a word problem that is unsolvable is to invent one whose solution would lead to a solution for the halting problem, which we know to be unsolvable. Specifically, we will show how to use a Turing–Post program P (which we assume consists of n instructions) to construct a word problem in such a way that a solution to the word problem we construct could be used to solve the halting problem for P. Therefore, if we begin with a program P whose halting problem is unsolvable, we will obtain an unsolvable word problem.

We will use an alphabet consisting of the $n + 4$ symbols:

$$1 \;\; 0 \;\; h \;\; q_1 \;\; q_2 \;\; \ldots \;\; q_n \;\; q_{n+1}.$$

The fact that the ith step of P is about to be carried out and that there is some given tape configuration is coded by a certain word (sometimes called a Post word) in this alphabet. This Post word is constructed by writing down the string of zeros and ones constituting the current nonblank part of the tape, placing an h to its left and right (as punctuation marks) and inserting the symbol q_i (remember that it is the ith instruction which is about to be executed) immediately to the left of the symbol being scanned. For example, with a tape configuration

$$\begin{array}{c} 11011 \\ \uparrow \end{array}$$

and instruction number 4 about to be executed, the corresponding Post word would be

$$h110q_411h.$$

This correspondence between tape configurations and words makes it possible to translate the steps of a program into equations between words. For example, suppose that the fifth instruction of a certain program is

PRINT 0.

We translate this instruction into the equations

$$q_40 \;=\; q_50, \qquad q_41 \;=\; q_50,$$

which in turn yield the equation between Post words

$$h110q_411h \;=\; h110q_501h$$

corresponding to the next step of the computation. Suppose next that the fifth instruction is

GO RIGHT.

It requires 6 equations to fully translate this instruction, of which two typical ones are

$$q_5 01 \;=\; 0q_6 1, \qquad q_5 1h \;=\; 1q_6 0h.$$

In a similar manner each of the instructions of a program can be translated into a list of equations. In particular when the ith instruction is **STOP**, the corresponding equation will be:

$$q_i \;=\; q_{n+1}.$$

So the presence of the symbol q_{n+1} in a Post word serves as a signal that the computation has halted. Finally, the four equations

$$\begin{aligned} q_{n+1}0 \;&=\; q_{n+1}, \qquad & q_{n+1}1 \;&=\; q_{n+1} \\ 0q_{n+1} \;&=\; q_{n+1}, \qquad & 1q_{n+1} \;&=\; q_{n+1} \end{aligned}$$

serve to transform any Post word containing q_{n+1} into the word $hq_{n+1}h$. Putting all of the pieces together we see how to obtain a word problem which "translates" any given Turing–Post program.

Now let a Turing–Post program P begin scanning the leftmost symbol of the string v; the corresponding Post word is hq_1vh. Then if P will eventually halt, the equation

$$hq_1vh \;=\; hq_{n+1}h$$

will be derivable from the corresponding equations as we could show by following the computation step by step. If on the other hand P will never halt, it is possible to prove that this same equation will not be derivable. (The idea of the proof is that every time we use one of the equations which translates an instruction, we are either carrying the computation forward, or—in case we substitute from right to left—undoing a step already taken. So, if P never halts, we can never get hq_1vh equal to any word containing q_{n+1}.) Finally, if we could solve this word problem we could use the solution to test the equation

$$hq_1vh \;=\; hq_{n+1}h$$

and therefore to solve the halting problem for P. If, therefore, we start with a Turing–Post program P which we know has an unsolvable halting

problem, we will obtain an unsolvable word problem.

Tape Configuration	*Program Step*
...001100...	1
↑	
...000100...	2
↑	
...000100...	4
↑	
...010100...	5
↑	
...010100...	7
↑	
...011100...	8
↑	
...011100...	1
↑	
...011000...	2
↑	
...011000...	2
↑	
...011000...	2
↑	
...011000...	4
↑	
...111000...	5
↑	
...111000...	5
↑	
...111000...	5
↑	
...111000...	7
↑	
...111100...	8
↑	
...111100...	10
↑	

Figure 2. Steps in a Computation by Doubling Program

Given an alphabet of three symbols a, b, c, and three equations

$$ba = abc$$
$$bc = cba$$
$$ac = ca$$

we can obtain other equations by substitution:

$$[ba]c = abcc$$

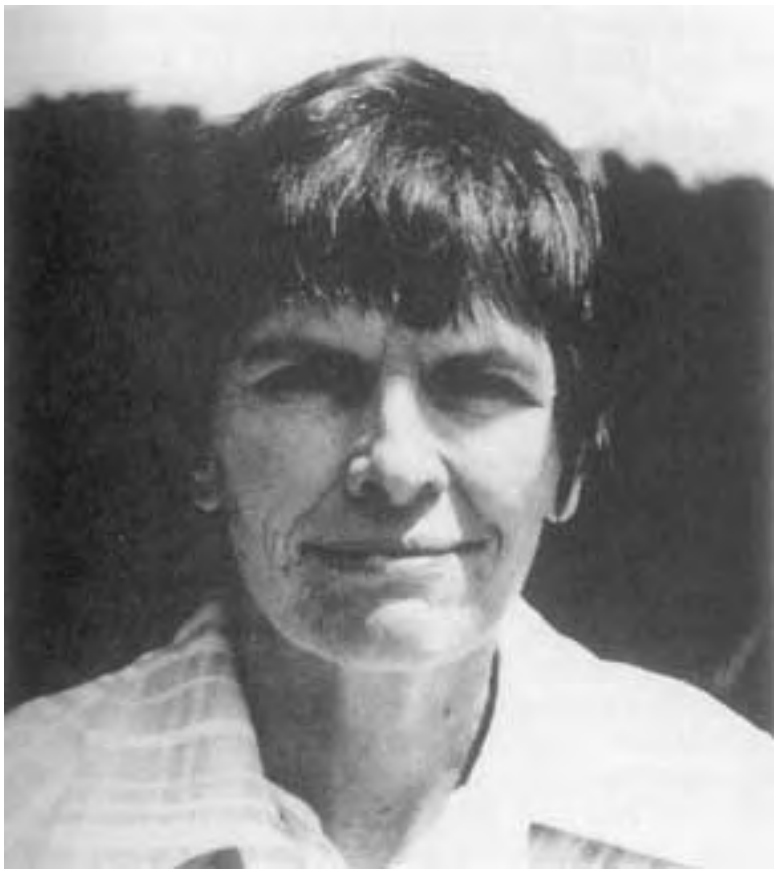
Or

$$\begin{aligned} b[ac] = [bc]a = c[ba]a = cabca = [ca]bca = acbca = \ldots \\ = cab[ca] = cabac = \ldots \\ = ca[bc]a = cacbaa = \ldots \end{aligned}$$

(The expressions in brackets are the symbols about to be replaced.) In this context can be raised questions such as: "Can we deduce from the three equations listed above that $bacabca = acbca$?" The word problem defined by the three equations is the general question: to determine of an arbitrary given equation between two words, whether or not it can be deduced from the three given equations.

Figure 3. A Word Problem

Julia B. Robinson



Julia B. Robinson was born in 1919 in St. Louis, Missouri, but has lived most of her life in California. Her education was at the University of California, Berkeley, where she obtained her doctorate in 1948. She has always been especially fascinated by mathematical problems which involve both mathematical logic and the theory of numbers. Her contributions played a key role in the unsolvability proof for Hilbert's tenth problem. In 1975 she was elected to the National Academy of Sciences, the first woman mathematician to be so honored.

```
 1.  PRINT 0
     ⋮
 9.  GO TO STEP 1 IF 1 IS SCANNED
10.  GO RIGHT
11.  GO TO STEP 22 IF 0 IS SCANNED
12.  GO RIGHT
13.  GO TO STEP 12 IF 1 IS SCANNED
14.  GO LEFT
15.  PRINT 0
16.  GO LEFT
17.  GO TO STEP 16 IF 1 IS SCANNED
18.  GO LEFT
19.  GO TO STEP 18 IF 1 IS SCANNED
20.  GO RIGHT
21.  GO TO STEP 1 IF 1 IS SCANNED
22.  STOP
```

Figure 4. A Program for Calculating Powers of 2

Chapter 6

On the Kolmogorov-Chaitin Complexity for short sequences

Jean-Paul Delahaye and Hector Zenil

Laboratoire d'Informatique Fondamentale de Lille
Centre National de la Recherche Scientifique (CNRS)
Université des Sciences et Technologies de Lille;
delahaye@lifl.fr, hector.zenil-chavez@malix.univ-paris1.fr

Among the several new ideas and contributions made by Gregory Chaitin to mathematics is his strong belief that mathematicians should transcend the millenary theorem-proof paradigm in favor of a quasi-empirical method based on current and unprecedented access to computational resources [3]. In accordance with that dictum, we present in this paper an experimental approach for defining and measuring the Kolmogorov-Chaitin complexity, a problem which is known to be quite challenging for short sequences — shorter for example than typical compiler lengths.

The Kolmogorov-Chaitin complexity (or algorithmic complexity) of a string s is defined as the length of its shortest description p on a universal Turing machine U, formally $K(s) = min\{l(p) : U(p) = s\}$. The major drawback of K, as measure, is its uncomputability. So in practical applications it must always be approximated by compression algorithms. A string is incompressible if its shorter description is the original string itself. If a string is incompressible it is said that the string is random since no patterns were found. Among the 2^n different strings of length n at least one will be completely random simply because there are not enough shorter strings. By using the same argument it can be also deduced that most of the strings have maximal K-C complexity. Therefore many of them will remain equal or very close to their original size after compression. Most of them will be therefore random. An important property of K is that it is nearly independent of the choice of U. However, when the strings are short in length, the dependence of K on a particular universal Turing machine U is higher producing arbitrary results. In this paper we will suggest an empirical ap-

proach to overcome this difficulty and to obtain a stable definition of the K-C complexity for short sequences.

Using Turing's model of universal computation, Ray Solomonoff [9, 10] and Leonid Levin [7] developed a theory about a universal prior distribution deeply related to the K-C complexity. This work was later known under several titles: universal distribution, algorithmic probability, universal inference, among others [5, 6]. This algorithmic probability is the probability $m(s)$ that a universal Turing machine U produces the string s when provided with an arbitrary input tape. $m(s)$ can be used as a universal sequence predictor that outperforms (in a certain sense) all other predictors [5]. It is easy to see that this distribution is strongly related to the K-C complexity and that once $m(s)$ is determined so is $K(s)$ since the formula $m(s)$ can be written in terms of K as follows: $m(s) \approx 1/2^{K(s)}$. The distribution of $m(s)$ predicts that non-random looking strings will appear much more often as the result of a uniform random process, which in our experiment is equivalent to running all possible Turing machines and cellular automata of certain small classes according to an acceptable enumeration. By these means, we claim that it might be possible to overcome the problem of defining and measuring the K-C complexity of short sequences. Our proposal consists of measuring the K-C complexity by reconstructing it from scratch basically approximating the algorithmic probability of strings to approximate the K-C complexity. Particular simple strings are produced with higher probability (i.e. more often produced by the process we will describe below) than particular complex strings, so they have lower complexity.

Our experiment proceeded as follows: We worked with Turing machines (TM) and cellular automata enumerations defined by Stephen Wolfram [11]. We let run (a) all $2-state\ 2-symbol$ Turing machines, and (b) a statistical sample of the $3-state\ 2-symbol$ ones, both henceforth denoted as $TM(2,2)$ and $TM(3,2)$.

Then we examine the frequency distribution of these machines' outputs performing experiments modifying several parameters: the number of steps, the length of strings, pseudo-random vs. regular inputs, and the sampling sizes.

For (a) it turns out that there are 4096 different Turing machines according to the formula $(2sk)^{sk}$ derived from the traditional $5-tuplet$ description of a Turing machine: $d(s_{\{1,2\}}, k_{\{1,2\}}) \rightarrow (s_{\{1,2\}}, k_{\{1,2\}}, \{1, -1\})$ where $s_{\{1,2\}}$ are the two possible states, $k_{\{1,2\}}$ are the two possible symbols and the last entry {1,-1} denotes the movement of the head either to the right or to the left. From the same formula it follows that for (b) there are 2985984 ma-

chines so we proceeded by statistical methods taking representative samples of size 1000, 5000, 10000, 20000 and 100000 Turing machines uniformly distributed over $TM(3,2)$. We then let them run 30, 100 and 200 steps each and we proceeded to feed each one with (1) a (pseudo) random (one per Turing machine) input string of length equal to the number of steps and (2) with a regular input.

We proceeded in the same fashion for all one dimensional binary cellular automata (CA), those (1) which their rule depends only on the left and right neighbours and those considering two left and one right neighbour, henceforth denoted by $CA(t,c)$[1] where t and c are the neighbour cells in question, to the left and to the right respectively. These CA were fed with (a) a single 1 surrounded by 0s and (b) a pseudo-random input string of length equal to the length . There are 256 one dimensional nearest-neighbour cellular automata or $CA(1,1)$, also called Elementary Cellular Automata (ECA) [11] and 65536 $CA(2,1)$. We then let them run 30, 100 and 200 steps each and we proceeded to feed each one with (1) a pseudo-random (one per cellular automata) input string of length equal to the number of steps and (2) with a regular input.

To determine the output of the Turing machines we look at the string consisting of all parts of the tape reached by the head. We then partitioned the output in substrings of length k. For instance, if $k = 3$ and the Turing machine head reached positions $1, 2, 3, 4$ and 5 and the tape contains the symbols {0,0,0,1,1} then the counter of the occurrences of the substrings 000, 001, 011 is incremented by one each. Similar for CA using the "light cone" of all positions reachable from the initial 1 in the time run. Then we perform the above for (1) each different TM and (2) each different CA, giving two distributions over strings of a given length k.

We then looked at the frequency distribution of the outputs of both classes TM and CA[2], (including ECA) performing experiments modifying several parameters: the number of steps, the length of strings, (pseudo) random vs. regular inputs, and the sampling sizes.

An important result is that the frequency distribution was very stable under the several variations described above allowing to define a *natural* distribution $m(s)$ particularly for the top of the distribution. We claim

[1]A better notation is the $3-$*tuplet* $CA(t,c,j)$ with j indicating the number of symbols, but because we are only considering $2-$*symbol* cellular automata we can take it for granted and avoid that complication.

[2]Both enumeration schemes are implemented in Mathematica calling the functions CelullarAutomaton and TuringMachine, the latter implemented in Mathematica version 6.0

that the bottom of the distribution, and therefore all of it, will tend to stabilise by taking bigger samples. By analysing 6.1 it can be deduced that the output frequency distribution of each of the independent devices of computation (TM and CA) follows an output frequency distribution. We conjecture that these systems of computation and others of equivalent computational power converge toward a single distribution when bigger samples are taken by allowing a greater number of steps and/or bigger classes containing more and increasingly sophisticated computational devices. Such distributions should then match the value of $m(s)$ and therefore $K(s)$ by means of the convergence of what we call their experimental counterparts $m_e(s)$ and $K_e(s)$. If our method succeeds as we claim, it could be possible to give a stable definition of the K-C complexity for short sequences independent of any constant.

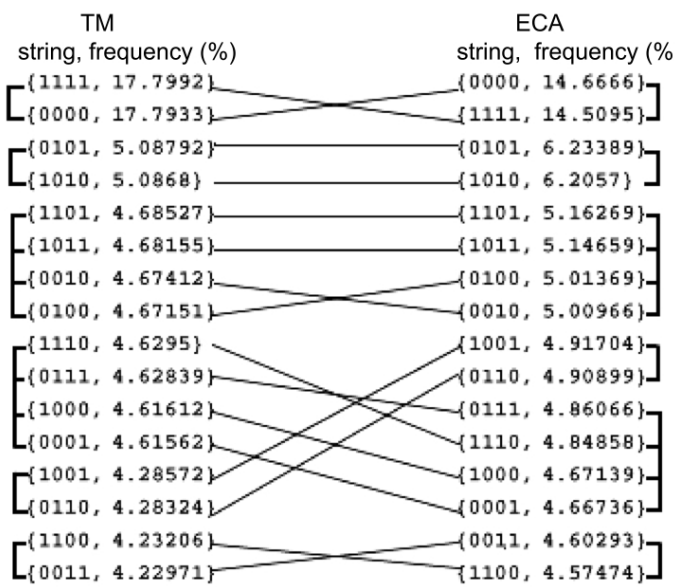


Figure 6.1. The above diagram shows the convergence of the frequency distributions of the outputs of TM and $ECA = CA(1, 1)$ for $k = 4$, after 200 steps, fed with (pseudo) random inputs. Matching strings are linked by a line. As one can observe, in spite of certain crossings, TM and ECA are strongly correlated and both successfully group equivalent output strings. By taking the six groups — marked with brackets — the distribution frequencies only differ by one.

For instance, the strings 0101 and 1010 are grouped in second place. They are therefore the second most complex group after the group composed by the strings of a sequence of zeros or ones but before all the other 2^k strings. And that is what it would be expected according to what algorithmic probability predicts since more structured non-random strings appear classified at the top (as our 0101... example) while less structured random-looking strings appear classified at the bottom. In favour of our claims about the nature of these distributions as following the universal distribution $m(s)$ and therefore approaching $K(s)$, notice that all strings were correctly grouped with their equivalent category of complexity under the three possible symmetries preserving their K-C complexity, namely reversion (sy), complementation (co) and composition of the two ($syco$). The fact that the method groups all the strings by their complexity category allowed us to apply a well-known lemma used in group theory to enumerate actual different cases, which let us consider only a single representative string for each of the complexity categories. For instance, for strings of length 10 ($k = 10$), the compressed distribution after the application of Burnside's lemma has 272 actual different strings from all $2^{10} = 1024$ original cases. The distribution below was built from $CA(3, 2)$ after 200 steps and regular inputs (a 1 surrounded by 0s). The following table contains the top 20 strings with their respective frequencies of appearance.

string	*frequency* (%)
0000000000	25.2308
0101010101	4.92308
0111101111	1.84615
0011110011	1.84615
0101001010	1.84615
0001001000	1.84615
0110000110	1.84615
0010001000	1.53846
0101101101	1.53846
0000100000	1.53846
0110111111	1.53846
0100100100	1.23077
0010010000	1.23077
0011111111	1.23077
0100100000	1.23077

0001000000 1.23077
0110001100 1.23077
0000110000 1.23077
0010110100 1.23077
0011100000 0.923077

Even though each distribution obtained by different means favoured different symmetries, it turned out that all them were strongly correlated to the others. Furthermore, we found that frequency distributions from several real- world data sources also approximates the same distribution, suggesting that they probably come from the same kind of computation, supporting contemporary claims about nature as performing computations [8, 11]. The extended paper available online contains more detailed results for strings of length $k = 4, 5, 6, 10$ as well as two metrics for measuring the convergence of $TM(2, 2)$ and ECA and the real-world data frequency distributions extracted from several sources.[3] Detailed papers with mathematical formulations and conjectures and the real-world data distribution results, are currently in preparation.

References

[1] G.J. Chaitin, *Algorithmic Information Theory*, Cambridge University Press, 1987.
[2] G.J. Chaitin, *Information, Randomness and Incompleteness*, World Scientific, 1987.
[3] G.J. Chaitin, *Meta-Math! The Quest for Omega*, Pantheon Books NY, 2005.
[4] C.S. Calude, *Information and Randomness: An Algorithmic Perspective (Texts in Theoretical Computer Science. An EATCS Series)*, Springer; 2nd. edition, 2002.
[5] Kirchherr, W., M. Li, and P. Vitányi. The miraculous universal distribution. *Math. Intelligencer* 19(4), 7-15, 1997.
[6] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, 1997.
[7] A.K.Zvonkin, L. A. Levin. "The Complexity of finite objects and the Algorithmic Concepts of Information and Randomness.", *UMN = Russian Math. Surveys*, 25(6):83-124, 1970.
[8] S. Lloyd, *Programming the Universe*, Knopf, 2006.
[9] R. Solomonoff, The Discovery of Algorithmic Probability, *Journal of Computer and System Sciences*, Vol. 55, No. 1, pp. 73-88, August 1997.

[3] It can be reached at `arXiv:http://arxiv.org/abs/0704.1043`. A website with complete results of the whole experiment is available at `http://www.mathrix.org/experimentalAIT`.

[10] R. Solomonoff, *A Preliminary Report on a General Theory of Inductive Inference*, (Revision of Report V-131), Contract AF 49(639)-376, Report ZTB-138, Zator Co., Cambridge, Mass., Nov, 1960
[11] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

Chapter 7

Circuit Universality of Two Dimensional Cellular Automata: A Review

Anahí Gajardo[α 1] and Eric Goles[β 2]

[α] *Departamento de Ingeniería Matemática, Universidad de Concepción, Casilla 160-C, Concepción, Chile;* anahi@ing-mat.udec.cl

[β] *Universidad Adolfo Ibáñez, Av. Diagonal Las Torres 2640, Peñalolén, Santiago, Chile*
Instituto de Sistemas Complejos de Valparaíso (ISCV), Av. Artillería 600B, C° Artillería, Valparaíso, Chile; eric.chacc@uai.cl

Universality of Cellular Automata (CA) is the ability to develop arbitrary computations, and is viewed as a "complexity certificate". The concept exists since the creation of CA by John von Neumann, and it has undergone several transformations and ramifications. We review a sample of models, starting with Banks's CA, where universality has been shown through the construction of arbitrary boolean circuits ("Circuit Universality"), in most but not all cases leading to proofs of Turing Universality.

7.1. Introduction

A d-dimensional Cellular Automata (d-CA) is a dynamical system evolving in $\mathbb{Z}^d$ in discrete time, where the upgrade of the lattice is synchronous and each site changes its state following a local rule which depends on the states of a fixed neighborhood.

Such a system can present very diverse and complex behaviors, the prediction of which is usually difficult. But what is exactly meant by this? If we completely know the initial configuration of a CA, we can *compute* its whole evolution up to any iteration t. On the other hand, if we only know the state of a finite part of the lattice, we can only update the state

[1] This work was partially supported by FONDECYT #1061036.
[2] The author is also affiliated with the Centro de Modelamiento Matemático (CMM), Universidad de Chile, UMR 2071-CNRS. This work was partially supported by FONDECYT #1070022.

of those cells whose neighbors's states are all known. In this case the set of cells that can be updated decreases over time until an instant in which we cannot compute anymore.

Let us illustrate this with an example in $\mathbb{Z}$. Consider the following rule: if the central cell is in state 1, it will change to 0 if any of its neighbors is in state 0; in any other case, it keeps its current state. (Figure 7.1 illustrates the evolution of this rule for a certain initial configuration). Let us now suppose that we know the initial state of cells 1 to 10. Initially, we can update only cells 2 to 9. In the next iteration we can update only cells 3 to 8, and, in general, at iteration i we can only update cells $1+i$ to $10-i$. At step 5, the state of every cell is unknown to us.

Figure 7.1. A space-time diagram of a 1-dimensional Cellular Automaton. Time evolves upward.

In general, in a d-dimensional CA with a neighborhood of radius 1, if we know the state of a hypercube of side $2t$, we can compute the state of the central cell for only $t-1$ iterations; the computation will take $O(t^{d+1})$ operations in a serial computer.

What means *to predict* in this context? It means to compute faster than that. In this context the previous CA is predictable: we can assert that the state of the central cell will be 1 at iteration $t-1$ if and only if we see no cell in state 0 at the beginning. Computing this last proposition takes only $O(t^d)$ on a serial computer, and $O(1)$ on a parallel computer with $O(t^d)$ processors. Thus, we can say that this CA is *simple*.

One may define the complexity of a CA rule f as the complexity of the following problem:

(CA-VALUE(f))[3] Given the configuration of a finite hypercube c of size *2t* and a given state s, decide whether after *t-1* steps, s will or will not be the state of the central cell of the hypercube, when the initial configuration is c and the CA rule is f.

Now, let us consider the class of polynomial problems (P), *i.e.*, problems

[3]this problem has been also called "CA-PREDICTION" by C. Moore.

which are solvable in polynomial time on a serial computer. Clearly the problem of predicting the state of a site at step t in a CA belongs to P. Within the class P, an important subclass is the class NC of problems that can be solved in polylogaritmic time on a parallel computer with a polynomial number of processors. It is not known whether P=NC or not; like in the case of the P=NP question, the concept of P-completeness plays an important role. P-complete problems are such that if any of them belongs to NC, then the complete class P is in NC, because every polynomial problem *reduces* to every P-complete problem. If, as is widely suspected, P≠NC, then being P-complete probably means to be *inherently secuencial*, out of NC. See R. Greenlaw *et al* [1], for example, for a detailed presentation of this subject.

An important P-complete problem [2] is CIRCUIT-VALUE:

(CIRCUIT-VALUE) Given a directed acyclic graph whose vertices have 0,1,2-ary logical gates, given a truth value assignment to each 0-ary vertex, and given a particular vertex called *output*, decide whether the output is or not True after computing each logical gate.

If we reduce CIRCUIT-VALUE to CA-VALUE(f), we prove that the second problem is P-complete. To achieve that for a particular 2-CA, it is enough to embed circuits in the lattice. In this case, the automaton is said to be *Circuit Universal*.

The technique of emulating a boolean circuit by a CA was first used by E. R. Banks [3] (1971). Banks embedded a complete computer design in the cellular lattice in order to prove that the CA was *Turing Universal*, *i.e.*, that it was able to simulate a Universal Turing machine (TM). Turing Universality implies Circuit Universality, since with a TM we can compute a circuit. The converse is not necessarly true, but it often turns out to work: in the case of Banks, his method to simulate circuits also allowed the simulation of a Turing machine, because a Turing machine can be computed with an infinite periodical circuit. See Z. Róka and B. Durand [4], N. Ollinger [5] and J-C. Delvenne *et al* [6] for extensive discussions of these and other notions of universality in Cellular Automata.

Evidently, circuit simulation is not the unique path to proving Turing Universality. One dimensional cellular automata, for instance, do not lend themselves easily to building logical circuits (though, if they are Turing Universal, we know that they can compute them). In particular, M. Cook showed the Turing universality of the elementary CA with Wolfram's code

110 by simulating a TAG-system [7, 8].

In this paper we will review several Circuit Universal models taken from different domains, thus giving a sample of different techniques. Some of these models are well known and/or important, arising mostly in physics and the field of artificial life; other have been chosen mostly for their technical interest.

7.2. Computing through signals

Turing Universality of two dimensional cellular automata has been proved mainly by emulating logical gates, in particular when working with a small number of states.

Roughly speaking, the idea is to represent logical values as 'signals' that travel in the space and interact with each other. The interactions must be rich enough to reproduce all the logical gates. Usually a signal represents the logical value *True*, and the absence of a signal is the logical value *False*. Logical gates are stable or periodical configurations, *i.e.*, finite configurations that, under certain boundary conditions, remain unchanged when iterating the CA rule, and if they change, they do it in a periodical way.

It is not necessary to exhibit each of the logical gates, a finite but complete set being enough. For example, it suffices to emulate a NOT and an AND gate, together with some devices allowing to direct, duplicate (a FANOUT) and to cross signals (a CROSSOVER) (Figure 7.2 shows a XOR gate constructed by pasting the needed pieces together). The CROSSOVER can be computed with a planar logical circuit by using only NOT and OR gates, and hence if we can emulate a NOT and an OR gate (or an AND gate) we do not need to give the CROSSOVER explicitly.

On the other hand, if we have the CROSSOVER, the AND gate and the OR gate are enough (the NOT gate is not necessary) because the MONOTONE CIRCUIT VALUE problem -which consists in computing a circuit with no NOT gates- is also P-complete [9].

With all the pieces in hand, a logical circuit is emulated by a configuration of the CA such that some cells represent the value of the input variables. When the CA evolves, the signals advance, enter the logical gates, and the circuit is computed. After a certain delay, one can recover the result of the computation by looking at whether a signal appears or not as the output of the last logical gate. We remark that in order to work correctly, signals must be synchronized: all the trajectories from the inputs

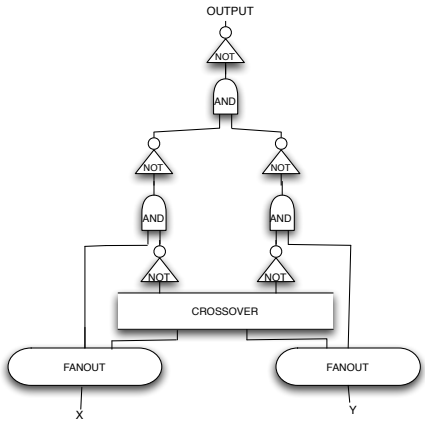


Figure 7.2. A scheme which, embedded in a CA, computes a XOR gate.

to the output must have the same length.

If the usual coding of True and False through presence and absence of signal is used, the NOT gate has some particular requirements. When a signal enters the NOT gate, nothing should exit; on the other hand, if no signal enters, the NOT gate must generate a signal. How will the NOT gate know when to generate this signal? This problem has been solved in two ways: 1) defining the NOT as a device that periodically emits signals when no signal enters; 2) allowing a second input to the NOT, which, in some sense, indicates when to compute.

The first solution has the inconvenient of periodically introducing signals in the system, thus producing many output signals before the "real" computed output. This is not as bad as it may sound, as long as we are able to discard all the spurious outputs and read only the good one. The second solution enlarges the amount of "wires" in the circuit, but only proportionally to the number of gates.

A an additional feature may be wanted in the gates and devices, though it is not really necessary it is used to ensure that information flows in the proper directions, *i.e.*, that gates do not send back signals in the direction of the inputs. This is achieved with a DIODE.

The first work that showed the universality of a CA by simulating a circuit was E. R. Banks's PhD thesis [3], which defined the smallest two-dimensional universal automaton: a CA with two states and von Neumann

neighborhood (*i.e.*, the four nearest cells in the 2-dimensional lattice). It is not easy to do this, and thus the Banks automaton with two states features some rather complicated devices. Banks also defined some three state automata, which are much simpler. In the next section we show one of them.

7.2.1. *A three states CA by Banks*

The local transition rule of Banks's CA is given in Figure 7.3. The figure shows the states of the cell together with its four nearest neighbors, followed by an arrow pointing to the state the cell will adopt in the next iteration. The rule is isotropic, and thus each configuration represents also its rotated versions. If a cell is in a situation not given in the list, then the cell does not change of state.

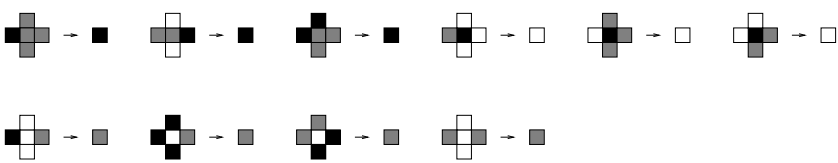


Figure 7.3. The transition rule of the three states Banks's CA.[5]

Banks defined a wire: a line of cells in gray state, embedded in a background of cells in blank state. This is a stable configuration, but if two neighboring cells in the chain are changed to blank and black, this perturbation propagates over the chain in the direction of the black cell (this is a signal), and if two signals collide, they disappear; a cut wire is a dead end for the signal. A junction of three or four wires is also a stable configuration; a grey cell is added in the three wires case, giving it a short cut wire. When one or two signals (in right angle) enter a junction they exit by the remaining wires. But if three signals enter a junction, they disappear. Figure 7.4 shows simulations of some of these phenomena.

The OR gate is simply a junction of three wires, which can be also used as a FANOUT. These gates can send signals in the direction of the inputs, something that Banks prefered to avoid. He managed this by defining a DIODE, shown in Figure 7.5(a), which only allows a signal to pass in one direction. Banks also constructed a *Timer* that generates signals periodically, shown in Figure 7.5(b): the signal inside the cycle duplicates each time it passes by the junction, sending a signal into the exiting wire. The

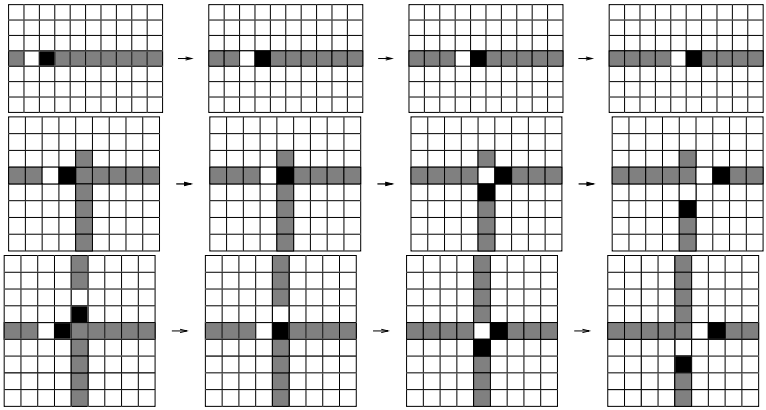


Figure 7.4. Top: A wire and the propagation of a signal. Middle: When one signal enters a three wire junction. Bottom: When two signals enter a four wire junction.

NOT gate is composed by two Timers connected to a junction (see Figure 7.5(c)). If a signal A arrives to this junction at the same iteration that both Timer signals, they mutually annihilates.

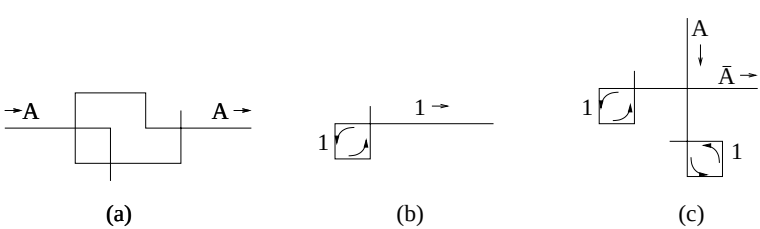


Figure 7.5. (a) the Diode. (b) the Timer. (c) the NOT gate.

Another universal CA was defined by B. Silverman [10], and is known as *Wireworld*. Its rule is very similar to those of Banks's CA, but it uses four states and Moore's neighborhood; this gives more flexibility and allows to build smaller and simpler circuits. In fact, a very small circuit that enumerates prime numbers has been constructed by D. Moore and M. Owen [11].

7.3. CA over a hexagonal grid and three states

The rule of this CA [12] by the present authors is very similar to those of Banks's CA. The fundamental difference is the cellular space: it works over the hexagonal grid, with the cells located on the vertices of the grid, and having thus only three neighbors each. The transition rule is as follows:

- blank state remains blank except if it has at least a black and a grey cells in its neighborhood, in which case it becomes grey, it also becomes gray if it has exactly two gray and one blank neighbors,
- grey state remains grey except if it has exactly one black neighbor, in which case it becomes black,
- black state always changes to blank state, except if it has a black neighbor, in which case it becomes grey.

In this CA we can define the following two configurations, which are enough to construct all the needed gates, as we will show.

Wire : a connected chain of grey cells over a background of blank cells. This configuration is stable, but if two neighboring cells in the chain are changed to blank and black, this perturbation propagates over the chain in the direction of the black cell (this is a signal, Figure 7.6(a)), and if two signals collide, they disappear.

Annihilating Junction : a device with three connected wires, such that: if it receives one signal on one of the wires, it generates one signal at each of the two remaining wires; but if it receives more than one signal simultaneously, it absorbs them and generates no outgoing signal (Figure 7.6(b)).

The Wire ensures that we can transport the signals anywhere in the cellular plane. The Annihilating Junction acts as a FANOUT when it receives only one signal. It can also act as a XOR gate when two of its wires are used as inputs, and this XOR can be used to construct a NOT gate, by taking the idea of Banks. As in Banks's work, a DIODE is required if we want to prevent a return of the signals back to the inputs. Since Banks's DIODE uses the four wires junction, its construction cannot be imported here. Figure 7.7(a) shows the solution, composed by two successive NOT gates. They will not change the input signals, but if a signal enters by the right, a triple collision is produced in the output of the second NOT; nothing changes inside the DIODE, and no signal goes to the left.

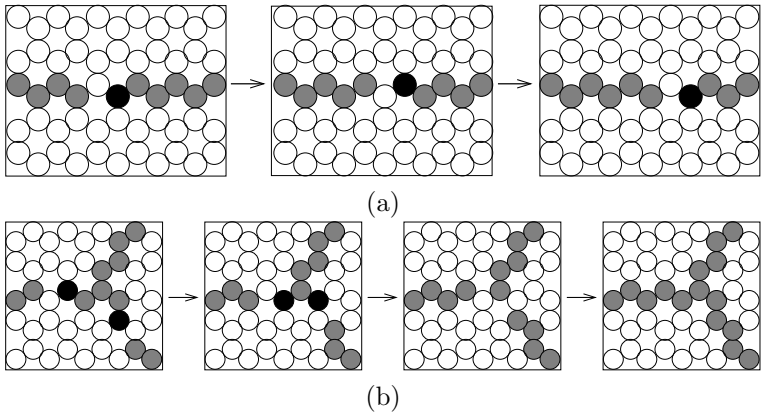


Figure 7.6. (a) A signal and the wire. (b) Two signals entering an Annihilating Junction.

Unfortunately, the XOR and the NOT gates are not enough for constructing the OR and the AND gates. In Banks's CA, the OR gate was obtained because two signals do not annihilate each other in a junction, something which does not work here. However, the logical gate 'p AND NOT q' can be computed within this system, as shown in Figure 7.7(b). In order to correctly use the gate, the input signals must be synchronized to collide at point C (or anywhere between the hexagon and the input p). If a signal arrives by p, it will exit. If a signal arrives by q, it will die at the Annihilating Junction A. If two signals arrive at the same time they will be mutually annihilated at point C and no output will be produced. Any logical gate can be constructed by using the 'p AND NOT q' gate, and hence the cellular automaton is Circuit Universal.

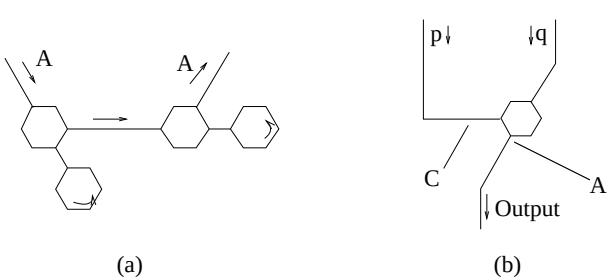


Figure 7.7. (a) The DIODE. (b) The p AND NOT q gate.

7.4. Life automata

7.4.1. *Game of life*

John Conway's famous *Game of Life* [13], which is known for its formidable variety of complex structures that make us think about life forms, is also universal. This automaton has two states (alive and dead) and Moore's neighborhood (*i.e.*, the eight nearest neighbors in the square grid). The rule is very simple: *a dead cell becomes alive if exactly three of its neighbors are alive, and an alive cell remains alive only if two or three of its neighbors are alive.*

Here the computation is made through signals that travel in the empty lattice, and the main logical gate is simply a collision. The Turing Universality of this automaton was proved by E. Berlekamp, J. H. Conway and R. Guy [13]. Here we will expose a variation of the proof by Z. Róka and B. Durand [4].

The signals are called *gliders*, one of which is shown in Figure 7.8(a) while travelling in the lattice. Notice that its displacement is done in four steps, implying that there are several kinds of collisions, depending on the respective phase of the participating gliders. One of these collisions kills both gliders (see Figure 7.8(b)).

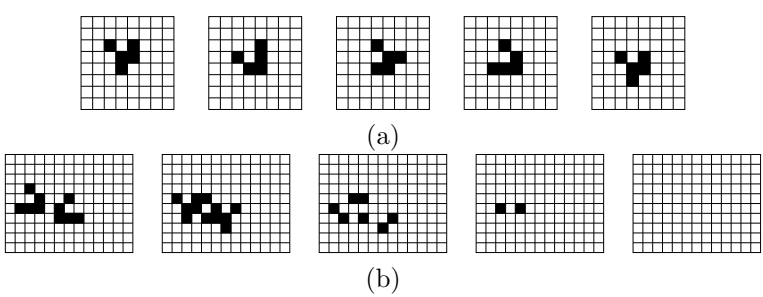


Figure 7.8. (a) A glider traveling in the lattice. (b) A glider crash.

This collision emulates a gate with two outputs: p AND NOT q and NOT p AND q, because each glider will continue its way if and only if the other glider does not arrive. It is possible to kill one of the outputs with a configuration called *eater*, which is a stable configuration that destroys the gliders when they collide with it. We obtain in this way a p AND NOT q gate. If we replace the input p by a periodical glider generator, which

exists (a *glider gun*), we obtain a NOT gate. With these two gates all the other gates can be obtained.

To complete the construction, we still need to duplicate and to direct the gliders, and this cannot be done with a logical gate. To achieve this, another collission must be used: a collision called *Kickback Collision*. When a stream of gliders collides with a glider in a Kickback collision, the first glider comes back to the stream and collides with the second glider. This last collision can kill the third glider too. In short, a single glider can kill three. This can be used to duplicate and change the direction of a logical value, as shown in Figure 7.9. The FANOUT requires several gliders (1's), which can be generated with glider guns.

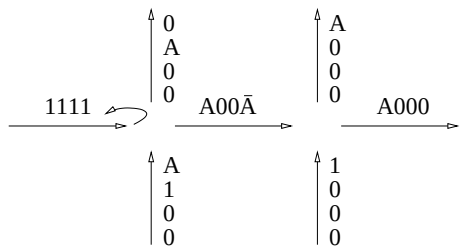


Figure 7.9. A FANOUT. If $A = 1$, the first three 1's of the left are killed, and the 1's coming from the bottom survive, as well as the last 1 which finally exits by the right. If $A = 0$, the last three 1's of the left are killed, and no glider goes up; only the first one goes on to kill the 1 coming from the right-bottom, preventing the exit of gliders in any direction.

7.4.2. *Life without death*

This CA has almost the same rule as the Game of Life, but in this case no cell ever dies. Here there are signals that travel in the lattice too, but they leave, of course, a trace of living cells. This phenomenon is called a *ladder*. The universality of this automaton was proved by D. Griffeath and C. Moore [14]. In their work, they establish three basic properties that the CA satisfies and that are enough to prove its P-completeness.

ladder : it is a configuration that grows in a straight line.
L : it is a stable configuration such that the ladder turns when colliding with it.
ladder collision : if one ladder collides with the trace of another ladder

it stops.

With the collision we have a p AND NOT q gate and all the other gates, including the CROSSOVER. If a ladder turns three times it collides with its own trace and stops, and this can be used to stop unused outputs. The FANOUT can also be constructed with this property (see Figure 7.10). Since within this system we have no 'ladder gun', all 1's must be produced at the beginning and delayed long enough to arrive at the desired iteration to the gate where they are needed. Griffeath and Moore proved that this can be done without making the circuit grow in an inconvenient way.

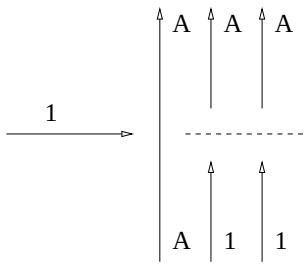


Figure 7.10. If a signal A passes before the 1 of the right, every other 1 passes. But if the signal A is 0, then the 1 of the right prevents the other 1's from going up.

7.5. Reversible models

Reversibility used to be seen as a limitation for universality. In fact, since the AND gate is not reversible, no reversible system can emulate an AND gate without producing some "garbage signals" (*i.e.*, additional outputs). For example, in the Game of Life, the basic gate p AND NOT q was obtained with a collision that generated two output signals and with an "eater" that killed unused output. In a Reversible Cellular Automata the equivalent of an "eater" cannot exist, and any "garbage signals" need to be "sent away" or "recycled." In 1982 E. Fredkin and T. Toffoli [15] formally proved that any boolean Circuit can be computed with only wires and a given reversible logical gate now known as *Fredkin Gate*. Their construction uses a linear number of input constants and does not generate additional "garbage signals." Later, in 1990, K. Morita [16] showed that a universal reversible 1-CA can also be computed with only wires and the Fredkin gate, showing in this way that in order to have Turing Universality in a 2-CA it is enough

to emulate wires and the Fredkin gate. Figure 7.11 shows the behavior of the Fredkin gate.

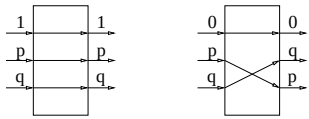


Figure 7.11. The Fredking gate is a three-inputs-three-outputs logical gate. If the upper input value is 1 (True), the outputs are equal to the inputs. If the upper input value is 0 (False), the upper output is also 0 but the other two inputs are permuted.

Two important reversible models are: 1) the Billiard Ball Model [15](BBM) (as well as its CA implementation [17]) and 2) the simulations of hydrodynamics and Navier-Stokes equations [18]. Essentially, both models represent a gas of identical hard spheres in a 2-dimensional space.

Let us start by studying the BBM. Figure 7.12 shows a balls collision. At time t we can put a ball either at X, Y, or both. If we put both balls, they will collide following the outer paths. Otherwise, one of the other paths will be followed. Then, with this collision, we have a two-inputs-four-outputs gate, where the outputs are: X AND Y twice, X AND NOT Y and NOT X AND Y. With an ingenious construction (Figure 7.13) this gate can be used to compute a Fredkin gate [15]. Hence the general Billiard Ball Model is P-complete and Turing Universal.

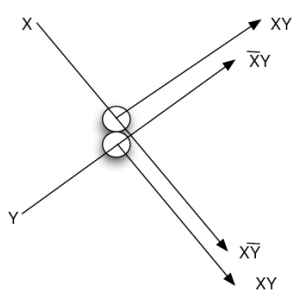


Figure 7.12. The collision of two balls can be viewed as a general logical gate with multiple outputs. The variables X and Y represent the presence or the absence of a travelling ball.

A sort of CA which simulates the BBM was defined by N. Margolus [17],

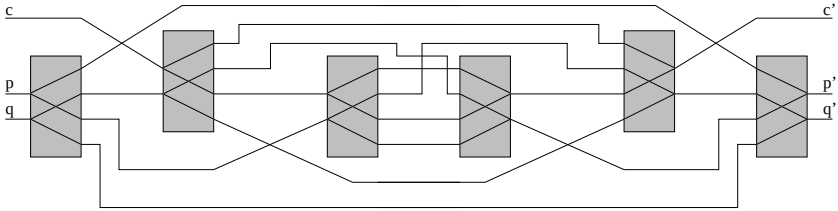


Figure 7.13. The Fredking gate computed only with the BBM collision. The gray boxes represent either the two-inputs-four-outputs gate given by the collision or its inverse. The lines outside these gates represent particle trajectories, which are controlled by well placed mirrors. The circuit must be constructed in such a way that the particles do not interact when their trajectories cross.

which he called *partitioned cellular automata* and have a special kind of iteration. Consider a 2-dimensional lattice with two states per site and divide the lattice in 2×2 blocks of sites. A transformation will be applied to each block. In our case the transformation is given in Figure 7.14. It is conservative and reversible. The updating consists in applying the rule alternately to the 2×2 blocks in the solid blocks partition and in the dotted one, as explained by Figure 7.15. One may see that this local rule allows to simulate the two dimensional BBM. Then, it is also Circuit Universal [17].

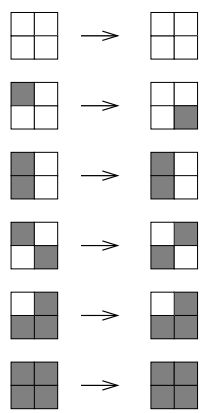


Figure 7.14. Rules for the BBM partitioned automata

Other reversible CA, also called 'partitioned CA' but in a different sense, are those defined by Morita *et al* [19–21]; they are proper CA. Morita and

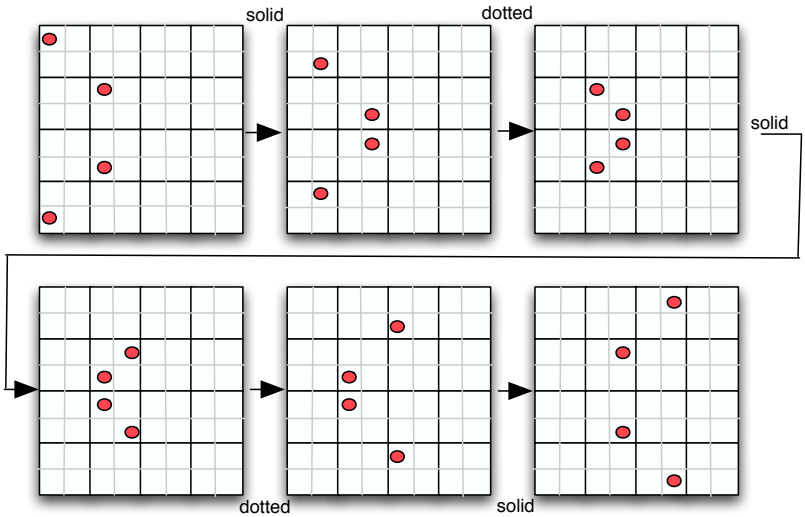


Figure 7.15. Billiard ball collision in the BBM automata.

his collaborators showed the universality of three simple 2-CA by emulating Fredking gates. Each publication cited does it for a different grid: a 32 states CA over the triangular grid, a 16 states CA over the square grid, and a 8 states CA over the hexagonal grid.

Lattice gas models are defined in a similar way as the BBM, also with collisions of particles. We will refer here to the simplest one, proposed by J. Hardy, O. de Pazzis and Y. Pomeau [18] in 1976. The HPP model considers only one kind of collision: when two particles collide head-on the outgoing particles are rotated by 90 degrees (see Figure 7.16). In any other case, the particles do not interact. This give us the same logical gate as the collision of the BBM, but here two of the outputs go back towards the inputs, requiring some tricky construction in order to recover them. The HPP lattice gas is P-complete, as was proved by C. Moore and M. Nordahl [22] in 1997. As an example, we give the construction of the NOT gate in Figure 7.17.

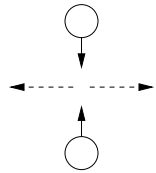


Figure 7.16. Two ball head-on collision in the HHP lattice gas.

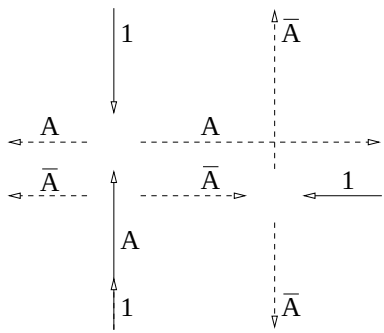


Figure 7.17. NOT gate for the HPP lattice gas. If a signal comes by A, it will collide with the 1 coming from above and two horizontal signals will exit. If no signal comes by A, the 1 from below will collide with the other one, then a signal will exit and collide the 1 from the right to finally exit by the top.

7.6. Sandpiles

Bak *et al* [23] introduced a model, mostly known as the Sandpile Automaton, which captures important features of many nonlinear dissipative systems. Consider a set of sites in a d-dimensional lattice such that each cell is connected to the $2d$ nearest neighbors. A finite number of tokens, $x_i(t) \geq 0$, is assigned to each cell i. Given the configuration at step t, if the site i has more than $2d$ tokens, it gives one of these to each neighbor. Since the updating is synchronous, the site i also receives one token from each neighbor with more than d tokens. In Figure 7.18 we give an example of the Sandpile dynamics in a two dimensional lattice.

If the number of tokens is finite, within a finite amount of time every site will have less than d tokens (see, for example, C. Moore [24] for a formal proof). Then, any finite configuration becomes stationary (a fixed point), that is to say, every avalanche stops in a finite number of iterations. The

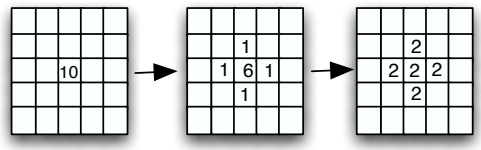


Figure 7.18. A simulation of the Sandpile.

interesting thing is that before reaching the equilibrium, the dynamics of this automaton is far from simple. In fact, for $d \geq 3$ and other topologies, it has been shown to be P-complete [24, 25]. It is not difficult to construct wires and some logical gates by using tokens: in Figure 7.19 we exhibit this constructions in dimension two [24].

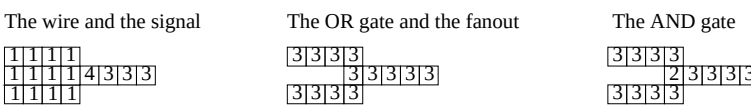


Figure 7.19. The logical gates for the sandpile model.

However, the construction of a CROSSOVER in two dimensions is not possible and this can be proved [26]; here we sketch the argument. A CROSSOVER can be viewed as a stable configuration in a finite portion of the 2-dimensional lattice. The circuit activity begins when some tokens are added to the input cells provoking avalanches. In order to cross avalanches, the CROSSOVER must be susceptible to develop a West-to-East and a South-to-North avalanche. The difficulty is that the avalanches intersect each other, because of the planarity of the grid, and the monotonicity of the rule makes one avalanche follow the course of the other one and vice versa. Figure 7.20 shows a configuration that could be a CROSSOVER. It can produce both West-to-East and a South-to-North avalanches. But when one of these avalanches is produced, tokens arrives both to the East and North sides.

The same result is obtained when we consider Moore's neighborhood (*i.e.*, the 8 neighbors of a site). But if we relax the size of the neighborhood further by considering radius 2, one can obtain a P-complete two dimensional model [26], whose logic gates as well as CROSSOVER are shown in

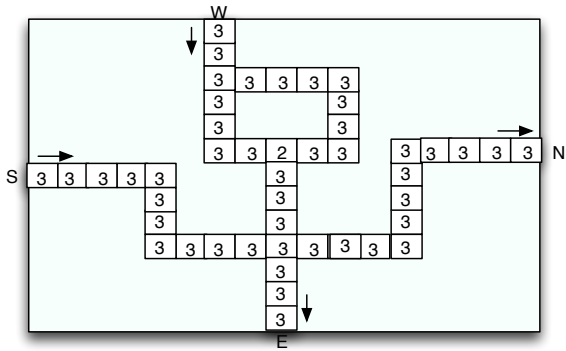


Figure 7.20. A failed CROSSOVER. If an avalanche comes in by the West or by the South, it exits both by the North and the East.

Figure 7.21.

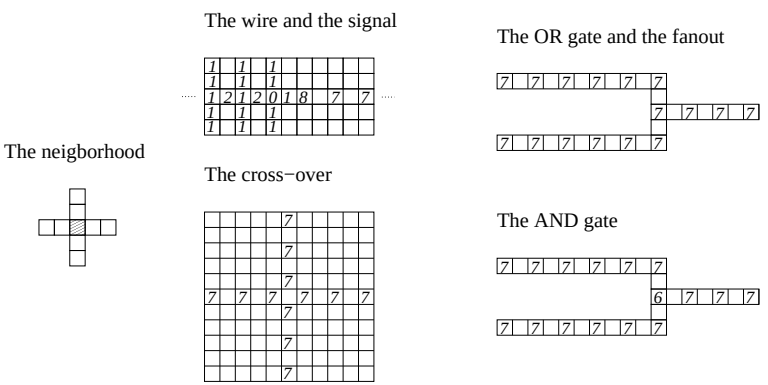


Figure 7.21. Logical gates constructed for a Sandpile with von Neumann neighborhood of radius two. In this model, the critical threshold is 8.

7.7. Conclusions

The construction of boolean circuits in 2-CA is not always an easy task, but it remains the best known path to proving Turing Universality. Turing Universality is an important property, since it means being able to perform any algorithmic calculation, and has strong consequences like the existence

of undecidable questions (in particular, that of the halting problem, which can be translated for a T.U. CA as the undecidability of knowing whether a certain local configuration will ever appear on the grid). But even when Turing Universality is not reached, *Circuit Universality* is by itself an interesting feature. If all the needed devices are available and can be properly combined, then configurations can be built to compute any given function $F : \{0,1\}^N \to \{0,1\}^M$; moreover, the P≠NC conjecture would put the dynamics of the CA in the "inherently sequential" category, making easy prediction hopeless. These two consequences of circuit construction are enough to justify it as a relevant certificate of complexity.

The set of models shows both the similarities and the diversity we can encounter when we try to follow Banks's steps toward universality. The shape and cardinality of the neighborhood can be a problem, sometimes solvable (like the case of the hexagonal grid, where it prevented an easy migration of some of Banks's constructions) and sometimes insurmountable (witness, the impossibility of crossovers in the standard sandpile). In some cases moving configurations are readily available (Life's gliders, billiard balls), but often a signal must be coded into a wire. The logic gates that are easier to build vary greatly between different automata too; and sometimes it is not in the logic gates, but in the complementary devices, where the most difficult part of the construction is found: duplicating, producing or even directing signals can be non-trivial, and even garbage disposal can become an issue.

Banks's approach or slight variants of it have even worked well in quite different settings, like Langton's ant [27], where changes on the lattice are made by the action of an agent (the ant) while the rest of the configuration remains static. Circuit and Turing universality were proved for this system by using gates and devices very much like those described here, the main differences being that there the configurations must be defined in such a way that the ant visits each gate to make it work, while logical values are "read" by the ant by affecting its trajectory, and "written" through the trail it leaves behind [28].

Are there precise features shared by all these automata, which allow them to be Circuit Universal? A set of sufficient conditions seems unlikely, but perhaps some necessary conditions could be found which the local rule of a 2-CA must satisfy if arbitrary boolean circuits are to be constructed. This is an intriguing open question, since proving non-universality is even more difficult, for non-trivial CA, than proving the opposite.

Acknowledgments

We want to thank particularly to Andrés Moreira for careful reading and precious comments.

References

[1] R. Greenlaw, H. Hoover, and W. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory.* (Oxford University Press, 1995).
[2] R. E. Ladner, The circuit value problem is log space complete for P, *SIGACT News.* **7**(1), 18–20, (1975).
[3] E. R. Banks. *Information Processing and Transmission in cellular automata.* PhD thesis, M.I.T., Cambridge, Mass., U.S.A., (1971).
[4] B. Durand and Z. Róka, *The game of life: universality revisited*, In eds. M. Delorme and J. Mazoyer, *Cellular Automata: a parallel model*, pp. 51–76. Kluwer Academic Pub., (1999).
[5] N. Ollinger. *Automates cellulaires : structures.* PhD thesis, École Normale Supérieure de Lyon, France, (2002).
[6] J.-C. Delvenne, P. Kůrka, and V. D. Blondel, Computational universality in symbolic dynamical systems, *Fundamenta Informaticae.* **74:4**, 463–490, (2006).
[7] M. Cook, Universality in elementary cellular automata, *Complex Systems.* **15**(1), 1–40, (2004).
[8] S. Wolfram, *A new kind of science.* (Wolfram Media Inc., 2002).
[9] L. M. Goldschlager, The monotone and planar circuit value problems are log space complete for P, *SIGACT News.* **9**(2), 25–29, (1977).
[10] A. K. Dewdney, Computer recreations: The cellular automata programs that create Wireworld, Rugworld and other diversions, *Scientific American.* **262(1)**, 146–149 (January, 1990).
[11] M. Owen, (2004). `http://www.quinapalus.com/wi-index.html`.
[12] A. Gajardo and E. Goles, Universal cellular automaton over a hexagonal tiling with 3 states, *International Journal of Algebra and Computation.* **11** (3), 335–354, (2001).
[13] E. Berlekamp, J. H. Conway, and R. Guy, *Winning Ways for Your Mathematical Plays.* vol. 2, (Academic Press., 1982).
[14] D. Griffeath and C. Moore, Life without death is P-complete, *Complex Systems.* **10**, 437–447, (1996).
[15] E. Fredkin and T. Toffoli, Conservative logic, *Int. J. of Theoret. Phys.* **21** (3/4), 219–253, (1982).
[16] K. Morita, A simple construction method of a reversible finite automaton out of Fredkin gates, and its related problem, *The Trans. of the IEICE.* **E73** (6), 978–984, (1990).
[17] N. Margolus. *Physics and Computation.* PhD thesis, M. I. T., (1987).
[18] J. Hardy, O. de Pazzis, and Y. Pomeau, Molecular dynamics of a classical

lattice gas: transport properties and time correlation functions, *Phys. Rev. A.* **13**, 1949–1960, (1976).

[19] K. Morita and S. Ueno, Computation-universal models of two-dimensional 16-state reversible cellular automata, *IEICE Trans. Inf. and Syst.* **E75-D** (1), 141–147, (1992).

[20] K. Morita, M. Margenstern, and K. Imai, Universality of reversible hexagonal cellular automata, *RAIRO Theoretical Informatics and Applications.* **33**, 535–550, (1999).

[21] K. Imai and K. Morita, A computation-universal two-dimensional 8-state triangular reversible cellular automaton, *Theoret. Comput. Sci.* **231**, 181–191, (2000).

[22] C. Moore and M. G. Nordahl. Predicting lattice gases is P-complete. Technical Report 034, Santa Fe Institute, Santa Fe, New Mexico (apr, 1997).

[23] P. Bak, C. Tang, and K. Wiesenfild, Self-organized criticality: An explanation of 1/f noise, *Phys. Rev. Lett.* **59**(4), 381–384, (1987).

[24] C. Moore and M. Nilsson, The computational complexity of Sandpiles, *J. of Stat. Phys.* **96**, 205–224, (1999).

[25] E. Goles and M. Margenstern, Sand pile as a universal computer, *Int. J. of Modern Physics C.* **7**(2), 113–122, (1996).

[26] A. Gajardo and E. Goles, Crossing information in two dimensional Sandpiles, *Theor. Comput. Sci.* **369**(1-3), 463–469, (2006).

[27] C. G. Langton, Studying artificial life with cellular automata, *Physica D.* **22**, 120–149, (1986).

[28] A. Gajardo, A. Moreira, and E. Goles, Complexity of Langton's ant, *Discrete Applied Mathematics.* **117**, 41–50, (2002).

Chapter 8

Chaitin's Graph Coloring Algorithm

James Goodman

Computer Science Department, University of Auckland, Auckland, New Zealand; goodman@cs.auckland.ac.nz

As a computer architect, and one oriented toward real-world problems, I am not familiar with many of Greg Chaitin's important contributions. I am aware of one very important one with practical impact: the Chaitin graph coloring algorithm for optimal global register allocation. This work, first disseminated beyond IBM in a 1981 publication [1], "Register Allocation via Coloring," demonstrated how a graph coloring algorithm could be solved in n^2 time to determine an optimal allocation of variables to registers. While the link between the four-color graph problem and register allocation had been recognized as early as 1957, graph coloring has long been known to be an NP-hard problem.

Chaitin's algorithm featured as a key innovation in the IBM 801 computer [2], the machine many would claim as the original RISC machine. Fran Allen points out [3] that prior to Chaitin's algorithm, "most global assignment methods were essentially variants on the FORTRAN I approach" [1954].

This algorithm was a major factor in achieving a goal expressed in the abstract of Chaitin's paper: "Preliminary results of an experimental implementation in a PL/I optimizing compiler suggest that global register allocation approaching that of hand-coded assembly language may be attainable." Global register allocation was an important step forward in computer architecture. With the technology advances of the 1980s, it was becoming increasingly clear that computers could be provided with far more general-purposes registers than the typical number of the day(8-16). But there was serious question about whether the registers could be adequately

utilyzed–hand-coded programs tested human limits by requiring the programmer to think about the efficient sharing of registers across different procedures that might or might not have concurrent scope. The development of an efficient global register allocation scheme was an important step forward, and justified the larger general register file typical of the new processors emerging in the early 1980s. Today compilers are able to do far better than humans can hope to do in allocating registers.

References

[1] G.J. Chaitin, M.A. Auslander, A.K. Chandra, J. Cocke, M.E. Hopkins, & P.W. Markstein. Register allocation via coloring, *Computer Languages*, Vol. 6, pp. 47–57, 1981.

[2] G. Radin. The 801 minicomputer, *IBM J. Res. Develop.*, Vol 27 (3), pp. 237–246, May 1983. The paper originally appeared in the *Proceedings of the (First) Symposium for Programming Languages and Operating Systems (ASPLOS)*, March 1982.

[3] F.E. Allen. The history of language processor technology in IBM, *IBM J. Res. Develop.*, Vol. 25 (6), pp. 535–548, September 1981.

Chapter 9

A Berry-type Paradox

Gabriele Lolli

Department of Mathematics
University of Torino, Italy; gabriele.lolli@unito.it

In this brief note we want to present and make known a statement of Berry paradox which has been ignored – buried in the graveyard of dead languages – and which is due to Beppo Levi, in 1908, independently from Russell. Berry paradox should actually be called Russell's paradox, as Chaitin has observed, in [3, pp. 8-9], on the basis of Alejandro Garciadiego's findings.

Beppo Levi's version is much more modern than Russell's informal one, since it is cast in arithmetical terms, with the appropriate numerical computations; although Levi's assessment of it is rather muddled, his paradox is ready for use when supplemented with the ideas which will transform the epistemological paradoxes in positive arguments in the theory of undecidable problems, as foreseen by Gödel and as realized in [2] or [1].

Beppo Levi doesn't mention Berry paradox and he doesn't cite the paper [10], where it was presented, which he quite certainly did not know; his only references are to [9]. He gives a mathematically rigorous version of the paradox in the course of a rather lengthy analysis of Richard's antinomy, where he follows and improves Peano's discussion of the latter.

1. Beppo Levi did not belong to Peano's school, although he graduated in Torino in 1896 with a dissertation in analysis. At the beginning of his career he worked in set theory, starting out with the perusal of Baire's thesis, then reverting to measure and category theory of the line after a failed attack at the continuum hypothesis. His name in mentioned in connection with the history of the axiom of choice, as he seems to have been the first to recognize (and formulate) the principle of partition in Berstein's proof that there are continuum many closed sets; he also gave a new proof avoiding

choice. Azriel Levy for example gives as reference for the axiom of choice "(Beppo Levi 1902; Erhard Schmidt 1904 – see Zermelo 1904)", in [6, p. 159], but Zermelo in 1904 only attributed, rightly, to Beppo Levi the principle of partition. Levi's precise contributions are spelled out in [8, pp. 78-80] and [7].

Further work by Beppo Levi in logic concerned mainly improvements and criticism of Peano's logic, and has only historical interest. As a mathematician he is best remembered for results in analysis related to Lebesgue theory, such as the theorem of the passage to limit under integral sign – equivalent to the later Lebesgue's dominated convergence – and in number theory, as explained in [11].

Beppo Levi had a crystal clear conception of the new axiomatic method as it had been recognized at the end of the nineteenth century; to him, on a par with Pasch, Hilbert, Enriques and Pieri, are due some of the most neat explications of the nature of mathematical theories, of the impossibility for primitive notions to be completely determined and of the unavoidability of multiple interpretations.

He relied on these principles also in the analysis of logical antinomies he gave in [4].

2. In discussing Richard's antinomy Levi denotes by E the set of real numbers definable with a finite number of words, and by R an enumeration of E.

He concludes that E does not exist, but the reason he gives is different from that of Poincaré and Russell imputing the antinomy to the impredicative character of the definition. He also notices the circularity of the definition of E, which to him, from his axiomatic point of view, means that E ed R are not to be considered defined and independent entities; they are primitive ideas subjected as a whole to some postulates, which he tries to identify:

(1) E is a set of numbers comprised between 0 and 1;
(2) R is an order of the numbers of E;
(3) All numbers of E and only they can be expressed with a finite number of symbols, including E and R.

Richard's diagonal number N proves only, according to Levi, that the postulates imposed on E and R are not mutually consistent.

Levi does not pinpoint the source of the contradiction, observing that "when a set of postulates is inconsistent, the contradiction lies in their union, not in any single one of them".

He dwells however on the ordering R, which is naturally thought of as obtained by a lexicographic ordering of the set F of all statements by pruning it of those which do not define a number or define a number previously listed. Let us call it briefly the lexicographic order.

Here comes the paradox (we had to change the counting with respect to Levi's because of the different length of sentences in Italian and in English).

> Let us call B, as we did, the number of symbols needed to compose our statements: among these symbols we'll assume to be included, for simplicity, the symbol for exponentiation $\uparrow$ ("to the"). It is easy to see that $B > 40$: suppose we write it in the decimal system, and let β be the number of its digits (we can reasonably assume $\beta = 2$). Now consider the proposition
>
> $$\ll \text{ The number whose place in } R \text{ is } B \uparrow B \gg \ ;$$
>
> this proposition contains $32 + 2\beta$ symbols[1]; its place in the enumeration of F is therefore $< B^{32+2\beta} < B \uparrow B$; if numbers of E are listed in the order described in n. 8 [by enumerating and pruning the members of F], the above defined number must have a place $< B \uparrow B$. *Hence that order cannot be R.*
>
> It is obvious that similar contradictions can be concocted for many other definitions of orderings one can imagine substituted to that of n. 8 [the lexicographic one]. Richard's contradiction presents itself for any way R can be thought to be defined[2] .

[1] [Blank spaces count as occurrences of a symbol for Levi.]

[2] We give also the italian original text for the historian's sake:

> Si chiami B, come pocanzi, il numero dei segni che servono a comporre le nostre frasi: fra questi segni supporremo compreso, per comodità, il segno $\uparrow$ (elevato a). Si vede facilmente che $B > 40$: lo supporremo scritto in cifre nel sistema decimale, e chiameremo β il numero di queste cifre (presumibilmente $\beta = 2$). Si consideri allora la proposizione
>
> $$\ll \text{ Il numero di posto } B \uparrow B \text{ in } R \gg;$$
>
> essa consta di $25+2\beta$ segni; il suo posto nell'ordinamento di F è quindi $< B^{25+2\beta} < B \uparrow B$; se dunque i numeri di E si numerano come si disse nel n. 8, il numero considerato dovrà avere posto $< B \uparrow B$. *Quella numerazione non può dunque essere R.*
>
> È chiaro che simili contraddizioni si possono costruire per molte al-

Levi is unaware that his argument can be made to stand independently of Richard's setting in the real numbers and of diagonalization, to which indeed it offers an alternative, while he continues to call it "Richard's antinomy". He could have noticed, with his axiomatic sensibility, that no mention is made of the nature of the definable entities, and that it could as well refer to the definability of natural numbers.

But it is clear that the general phenomenon upon which Levi stumbled is that of descriptions which are shorter than what they describe. What is missing of course, to arrive at an instance of the incompleteness phenomenon, is the idea of proving in a formal system which is the place of the number defined as "the number whose place is $B \restriction B$". Since this description is shorter than $B \restriction B$, the number defined should occupy a place $< B \restriction B$, so no place can be proved to exist for it.

What Beppo Levi instead argues is the following, as can be inferred by his last vague comment: think of the order R as the lexicographic order, then the number "whose place is $B \restriction B$" shows that the order cannot be that one. But think of R as any other ordering, and a similar argument will show that the order cannot be the one you thought. So there is none, which is consistent with E.

References

[1] G. Boolos, "A new proof of the Gödel incompleteness theorem", *Notices AMS*, **36**, pp. 388-90.
[2] G. J. Chaitin, "The Berry paradox", *Complexity*, **1**, n. 1 (1995), pp. 26-30.
[3] G. J. Chaitin, *The Unknowable*, Springer, Singapore, 1999.
[4] B. Levi, "Antinomie logiche?", *Annali di Matematica Pura e Applicata*, (III) (1908), XV, pp. 187-216, in [5, vol. 2, pp. 629-58].
[5] B. Levi, *Opere 1897-1926*, 2 voll., edited by Unione Matematica Italiana, Cremonese, Roma, 1999.
[6] A. Lévy, *Basic Set Theory*, Springer, Berlin, 1979.
[7] G. Lolli, "L'opera logica di Beppo Levi", in [5, vol. 1, pp. LXVII-LXXVI].
[8] G. H. Moore, *Zermelo's Axiom of Choice*, Springer, New York, 1982.
[9] B. Russell, *The principles of mathematics I*, Cambridge Univ. Press. Cambridge, 1903.

---

tre definizioni di numerazioni che si vogliano immaginare sostituite a quella del n. 8. La contraddizione del Richard si presenta comunque R si voglia immaginar determinata.

[4, p. 638].

[10] B. Russell, "Les paradoxes de la logique" *Revue de métaphysique et de morale*, **14**, pp. 627-50.

[11] N. Schappacher and R. Schoof, "Beppo Levi and the arithmetic of elliptic curves", *The Mathematical Intelligencer*, **18**, winter 1996, pp. 57-68.

Chapter 10

Ω in Number Theory

Toby Ord[α] and Tien D. Kieu[β]

[α] *Faculty of Philosophy, University of Oxford, Oxford, OX1 4JJ, UK;* toby.ord@philosophy.ox.ac.uk

[β] *Centre for Atom Optics and Ultrafast Spectroscopy, Swinburne University of Technology, Hawthorn 3122, Australia;* kieu@phg.com.au

We present a new method for expressing Chaitin's random real, Ω, through Diophantine equations. Where Chaitin's method causes a particular quantity to express the bits of Ω by fluctuating between finite and infinite values, in our method this quantity is always finite and the bits of Ω are expressed in its fluctuations between odd and even values, allowing for some interesting developments. We then use exponential Diophantine equations to simplify this result and finally show how both methods can also be used to create polynomials which express the bits of Ω in the number of positive values they assume.

10.1. Recursive Enumerability, Algorithmic Randomness and Ω

One of the most startling recent developments in the theory of computation is the discovery of the number Ω, through the subfield of algorithmic information theory. Ω is a real number between 0 and 1 which was introduced by G. J. Chaitin [2] as an example of a number with two conflicting properties: it is both recursively enumerable and algorithmically random. Very roughly, this means that Ω has a simple definition and can be computed in the limit from below, yet we can determine only finitely many of its digits with certainty—for the rest we can do no better than random.

Understanding the full importance of these properties requires some familiarity with the recursive functions—commonly presented through models of computation such as Turing machines or the lambda calculus. For the purposes of algorithmic information theory, however, it is convenient to

abstract some of the details from these models and consider a programming language in which the (partial) recursive functions are represented by finite binary strings.[1] These strings are just programs for a universal Turing machine (or universal lambda expression) and they take input in the form of a binary string then output another binary string or diverge (fail to halt). For convenience, we will often consider these inputs and outputs to encode tuples of positive integers.

On top of this simplified picture of computation, we impose one restriction which is necessary for the development of algorithmic information theory (and hence Ω). The set of strings that encode the recursive functions must be prefix-free. This means that no program can be an extension of another, and thus each program is said to be self-delimiting. As algorithmic information theory is intricately linked with communication as well as computation, this is quite a natural constraint—if you wish to use a permanent binary communication channel, then you need to know when the end of a message has been reached and this cannot be done if some messages are extensions of others.

There are many prefix-free sets that one could choose and many recursive mappings between these and the recursive functions. These different choices of 'programming language' lead to different values of Ω, but this does not matter much as almost all of its significant properties will remain the same regardless. However, to allow talk of Ω as a specific real number we will use the same language as Chaitin [3].

Now that we have explained what we mean by a programming language, we can give a quick overview of computability in terms of programs. A program computes a set of n-tuples if, when provided with input $\langle x_1, \ldots, x_n \rangle$, it returns 1 if this is a member of the set and 0 otherwise. A program computes an infinite sequence if, when provided with input n, it returns the value of the n-th element in the sequence. A program computes a real, r, if it computes a sequence of rationals $\{r_n\}$ which converges to r and $|r - r_n| < \frac{1}{2^n}$. These sets, sequences and reals that are computed by programs are said to be recursive.

There are also many sets, sequences and reals that cannot be computed, but can be approximated in an important way. A program semi-computes a set of n-tuples if, when provided with input $\langle x_1, \ldots, x_n \rangle$, it returns 1 if this is a member of the set and diverges otherwise. A program semi-computes

[1] For more details see Chaitin [3].

an infinite sequence of bits if, when provided with input n, it returns 1 if the n-th bit in the sequence is 1 and diverges otherwise. A program semi-computes a real, r, if, when provided with input n, it computes a rational number, r_n, where $\{r_n\}$ converges to r from below. These sets, infinite bitstrings and reals that are semi-computed by programs are said to be recursively enumerable or r.e.

There is an important point that needs to be made concerning reals and their representations. Each real number between 0 and 1 has a binary expansion: a binary point followed by an infinite sequence of bits that represents the real.[2] Throughout this paper, we shall be making considerable use of the binary expansions of real numbers so it is important to point out an oddity in the definitions above: a real is recursive if and only if its binary expansion is recursive, but a real may be r.e. even if its binary expansion is not r.e. We shall thus take care to distinguish the weaker property of being an *r.e. real* from the stronger one of being a *real whose binary expansion is r.e.*

An example of a real that is r.e. but not recursive is τ: the real number between 0 and 1, whose k-th digit is 1 if the k-th program (in the usual lexical ordering of finite bitstrings) halts when given the empty string as input and 0 if the k-th program diverges. Equivalently:

$$\tau = \sum_{p_n \text{ halts}} 2^{-n} \tag{10.1}$$

τ is an r.e. real because there is a computable sequence of rationals $\{\tau_i\}$, where

$$\tau_i = \sum_{\substack{n \leq i \\ p_n \text{ halts in } \leq i \text{ steps}}} 2^{-n} \tag{10.2}$$

such that $\{\tau_i\}$ converges to τ from below.

Furthermore, it is clear that the binary representation of τ is also r.e. because there is a program that simulates the k-th program, halting if and only if it does. This program is a slightly modified universal program that first determines the bits of the k-th program and then simulates it.

[2]For numbers that can be expressed with a representation ending in an infinite string of 0's, there is another representation ending in an infinite sequence of 1's, but we shall remove this ambiguity by only using representations with an infinite number of 0's. This will not affect the important reals in this paper, Ω and τ, as they are irrational and thus have unique representations regardless.

τ is not recursive, however, because if a program could compute it to arbitrary accuracy, it would determine whether each program halts or not when given the empty string as input. This is known as the *blank tape problem* and is easily shown to be equivalent to the more general *halting problem*—'does a given program halt on a given input?'. The halting problem is fundamental to the theory of computation and is the most famous problem that cannot be recursively solved. τ merely encodes the information necessary to solve the halting problem into the binary expansion of a real number and thus provides a very simple example of a non-computable real to which we can contrast the more exotic properties possessed by Ω.

Ω encodes the halting problem in a more subtle way: it is the *halting probability*. We could, theoretically, generate a random program one bit at a time, by flipping a fair coin and writing down a 1 when it comes up heads and a 0 for tails—stopping if we reach a valid program. The chance of generating any given n bit program is therefore $\frac{1}{2^n}$. Ω is the chance that this method of random program construction generates a program that halts. Letting $|p|$ represent the size of p in bits, we can also express Ω as

$$\Omega = \sum_{p \text{ halts}} 2^{-|p|} \tag{10.3}$$

As was the case for τ, there is a computable sequence of rationals $\{\Omega_i\}$, where

$$\Omega_i = \sum_{\substack{|p| \leq i \\ p \text{ halts in} \leq i \text{ steps}}} 2^{-|p|} \tag{10.4}$$

which converges to Ω from below, showing it to be an r.e. real. However, we shall see shortly that the binary representation of Ω is *not* r.e.

A real is said to be *algorithmically random* [3] if and only if the 'algorithmic complexity' of each n-bit initial segment of its binary expansion becomes and remains arbitrarily greater than n.[3] In other words a real, r, is algorithmically random if and only if any program that has access to outside advice in the form of binary messages requires more than n bits of advice to compute the first n bits of r's binary expansion (for all values of n above some threshold).[4] Thus a random real is one for which only finitely

[3]This is only one of four common definitions of algorithmic randomness, however, all have been shown to be equivalent.

[4]The reason that slightly more than n bits of advice are needed is because in algorithmic information theory the advice comes in self-delimiting messages (which are actually programs that generate the advice—like self-extracting archives) and in order to be self-

many prefixes of its binary expansion can be compressed.

It is easy to see that a random real cannot have an r.e. binary expansion. Let x be an arbitrary real whose binary expansion is r.e. By definition, there must be a program, p_x, that takes a positive integer, k, and halts if and only if the k-th bit of x is 1. To determine n bits of x, we just need to know how many of these n values of k make p_x halt. We could then simply run p_x on all the values of k and stop when this many have halted, knowing that no more will halt and thus determining the n bits of x. Since all positive integers less than n can be encoded in $\log n$ bits (rounding up), we only need to send a message of about $(\log n + \log\log n)$ bits. In this manner, any prefix of x can be significantly compressed, so x cannot be random.

Because of this, we can see that τ too is not random. However, Chaitin [3] has proven that Ω *is* random and so cannot be compressed in this manner.[5] For sufficiently high values of n, n bits of Ω provide n bits of algorithmically incompressible information.

In addition to recursive incompressibility, random reals are also characterised by *recursive unpredictability* [3]. Consider a 'predictive' program that takes a finite initial segment of an infinite bitstring and returns a value indicating either 'the next bit is 1', 'the next bit is 0' or 'no prediction'. If any such program is run on all finite prefixes of the binary expansion of a random real and makes an infinite amount of predictions, the limiting relative frequency of correct predictions approaches $\frac{1}{2}$. In other words when any program is used to predict infinitely many bits of a random real, such as Ω, it does no better than random—even with information about all the prior bits.

The power of this unpredictability can be seen when compare the predictability of τ. In this case, the predictive program can easily predict an infinite amount of bits with no errors. This is because infinitely many bits of τ are 'easy' to compute. For example, consider the halting behaviour of Turing machines: there are infinitely many Turing machines which have no loops in their transition graphs and thus cannot possibly diverge. When the predictive program is asked to predict the n-th bit of τ, it can just

delimiting, these messages need slightly more bits than they would otherwise. In general, an n bit string requires about $(n + \log n)$ bits. Chaitin [3] provides further details.

[5]Indeed, it has since been shown through the work of R. Solovay, C. S. Calude, P. Hertling, B. Khoussainov, Y. Wang and T. A. Slaman that the only r.e. random reals are Ω's for different programming languages. See Calude [1] for more details.

check to see if the n-th program corresponds to such a machine, returning 'the next bit is 1' if it does and 'no prediction' otherwise.[6]

With its inherent incompressibility and unpredictability, Ω really does go beyond the type of uncomputability present in a more typical non-recursive real such as τ. However, its contrasting property of being an r.e. real makes Ω seem to be just beyond our reach. In the next section, we will introduce Diophantine equations and show how these can be used to bring uncomputability into the more classical field of number theory. Then, in Section 10.3, we will show two ways of using Diophantine equations to bring Ω and randomness to number theory—Chaitin's original method and our new technique.

10.2. Diophantine Equations and Hilbert's Tenth Problem

A Diophantine equation is a polynomial equation in which all of the coefficients and variables take only positive integer values. Many natural phenomena with discrete quantities are modelled well by Diophantine equations and they occur frequently in number theory. It is often convenient to express a Diophantine equation with all terms on the left hand side:

$$D(x_1, \ldots, x_m) = 0 \tag{10.5}$$

Here D is a polynomial of $x_1, \ldots, x_m$ in which the coefficients can take both positive and negative integer values.

The number of solutions for a Diophantine equation varies widely. For example, $3x_1 + 6 = 0$ has one solution, while $x_1x_2 - 2 = 0$ has two and $x_1x_2 - x_2 = 0$ has infinitely many. Some however, such as $2 - 3x_1 = 0$, have no solutions at all. There are many different methods for deciding whether Diophantine equations of certain forms have solutions and determining what these solutions are, but there has been a great desire for a single method that takes an arbitrary Diophantine equation and determines whether or not it has solutions. In 1900, David Hilbert [6] gave the problem of finding such a method as the tenth in his famous list of important problems to be addressed by mathematicians in the 20th Century. Since then, the task of finding this method has become known simply as Hilbert's Tenth Problem.

[6]From the definition of binary programs in algorithmic information theory, there must be a recursive mapping between programs and Turing machines (or any such model).

Another area of research concerns families of Diophantine equations. A family of Diophantine equations is a relation of the form:

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0 \tag{10.6}$$

in which we distinguish between two types of variable. The variables $x_1, \ldots, x_m$ are called unknowns, while $a_1, \ldots, a_n$ and called parameters. By assigning values to each of the parameters (and treating them as constants), we pick out an individual Diophantine equation from the family. For example, the family $a_1 - 3x_1 = 0$ consists of the equations: $1 - 3x_1 = 0$, $2 - 3x_1 = 0$, $3 - 3x_1 = 0$ and so on.

Each family of Diophantine equations is naturally associated with a certain set of n-tuples of positive integers, $\mathfrak{D}$, in the following manner:

$$\langle a_1, \ldots, a_n \rangle \in \mathfrak{D} \iff \exists x_1 \ldots x_m D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0 \tag{10.7}$$

In other words, a tuple is in the set if the equation it corresponds to has a solution. Such sets are said to be Diophantine or to have a Diophantine representation. For example, the set of all multiples of 3 is Diophantine because it is represented by the family $a_1 - 3x_1 = 0$.

Over the 1950's and 1960's, M. Davis, H. Putnam and J. Robinson established several important results regarding which sets are Diophantine. Their key result concerned a characterisation, not of Diophantine sets, but their close relation: *exponential* Diophantine sets.

A family of exponential Diophantine equations is a relation of the form:

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m, 2^{x_1}, \ldots, 2^{x_m}) = 0 \tag{10.8}$$

where D is once again a polynomial, but now some of its variables are exponential functions of others. Davis, Putnam and Robinson [5] used this additional flexibility to show that all r.e. sets are exponential Diophantine. It had long been known that all exponential (and standard) Diophantine sets are r.e. because it is trivial to write a program that searches for a solution to a given equation and halts if and only if it finds one. Therefore, the new result meant that the exponential Diophantine sets were precisely the r.e. sets.

In 1970, Yu. Matiyasevich [7] completed the final step, proving that all exponential Diophantine sets are also Diophantine and thus that the Diophantine sets are exactly the r.e. sets—a result now known as the DPRM Theorem.

The DPRM Theorem provides an intimate link between Diophantine equations and computability, reducing the task of determining whether a set has a Diophantine representation to a matter of programming. For instance, there is a program that takes a single input k and halts if and only if the k-th bit of τ is 1. Thus, the set of positive integers that includes k if and only if the k-th program halts is an r.e. set and via the DPRM Theorem, there is a family of Diophantine equations with a parameter k, that has solutions if and only if the k-th program halts.

This family of equations provides an example of uncomputability in number theory and shows that Hilbert's Tenth Problem must be recursively undecidable because a program that finds whether arbitrary Diophantine equations have solutions could be used to determine the bits of τ and thus to solve the halting problem. Indeed, it was long known that the recursive undecidability of Hilbert's Tenth Problem would follow immediately from the DPRM Theorem and this was the main motivation for its proof—the Diophantine representations for all other r.e. sets being largely a bonus.

10.3. Expressing Omega Through Diophantine Equations

While the DPRM Theorem demonstrates the existence of τ and uncomputability in number theory, it also denies the possibility of finding a similar family of Diophantine equations expressing Ω and randomness. This is due to the fact discussed in Section 10.1 that, while Ω is an r.e. real, its sequence of bits is *not* r.e. However, the DPRM Theorem only prohibits a direct Diophantine representation of Ω and says nothing about the more subtle properties of Diophantine equations in which these bits could perhaps be encoded.

Chaitin [3] takes such an approach. While there is no program of one variable, k, that halts if and only if the k-th bit of Ω is 1, Chaitin provides a program, P, that takes two variables, k and N, and computes Ω somewhat less directly. For a given value of k, P can be thought of as making an infinite series of 'guesses' as to the value of the k-th bit of Ω—when P is run on k and N, it gives the N-th guess as to the k-th bit of Ω. What is impressive is that P gets infinitely many of these guesses right and only finitely many wrong.

How does P do this? It simply computes the sequence $\{\Omega_i\}$ discussed in Section 10.1 until it gets to Ω_N and then returns the k-th bit of Ω_N. Just as $\{\Omega_i\}$ forms a sequence of approximations to Ω, so the k-th bit of

each $\{\Omega_i\}$ forms a sequence of approximations to the k-th bit of Ω.

Consider this k-th bit of each $\{\Omega_i\}$ as i is increased. This bit could change between 0 and 1 many times, but since $\{\Omega_i\}$ approaches Ω, it must eventually remain fixed, at which point it must have the same value as the k-th bit of Ω. Therefore, if the k-th bit of Ω is 1, the k-th bit of $\{\Omega_i\}$ must be 0 for only finitely many values of i, and so P must return 0 for finitely many values of N and 1 for infinitely many. On the other hand, if the k-th bit of Ω is 0, then the k-th bit of $\{\Omega_i\}$ must be 1 for only a finite number of values of i and P must return 1 for finitely many values of N and 0 for infinitely many. Either way, as N increases, the output of P applied to k and N limits to the k-th bit of Ω.

It may seem as though this program is computing the bits of Ω but this is not quite the case. P just computes the N-th 'guess' of the k-th bit. From the infinite sequence of such guesses, the k-th bit could be determined but P does not and cannot put the guesses together like that—it just returns one of them.

Since recursive functions are just a special type of r.e. function, we can apply the DPRM Theorem and see that there must be a family of Diophantine equations

$$\chi_1(k, N, x_1, \ldots, x_m) = 0 \tag{10.9}$$

that has solutions for given values of k and N if and only if P returns 1 when provided with these as input. For a given value of k, there are solutions for infinitely many values of N if and only if the k-th bit of Ω is 1.

Thus, by using a more subtle property of the family of Diophantine equations, Chaitin was able to show that algorithmic randomness occurs in number theory: as k is varied, there is simply no recursive pattern to whether this family of equations has solutions for finitely or infinitely many values of N.

By modifying Chaitin's method slightly, we can find a new way of expressing the bits of Ω through a family of Diophantine equations. We will present this method informally here, with complete details being found in [9] (see also [4]). Our result has now been extended by Matiyasevich [8].

Consider a new program, Q, that also takes inputs k and N, and begins to compute the sequence $\{\Omega_i\}$. For each value of Ω_i, Q checks to see if it is greater than $\frac{N}{2^k}$, halting if this is so, and continuing through the sequence otherwise. Since $\{\Omega_i\}$ approaches Ω from below, we can see that $\Omega_i > \frac{N}{2^k}$

implies that $\Omega > \frac{N}{2^k}$ and conversely, if $\Omega > \frac{N}{2^k}$ there must be some value of i such that $\Omega_i > \frac{N}{2^k}$. Therefore, Q will halt on k and N if and only if $\Omega > \frac{N}{2^k}$. Alternatively, we could say that Q recursively enumerates the pairs $\langle k, N \rangle$ such that $\Omega > \frac{N}{2^k}$.

Just as we could determine the k-th bit of Ω from the number of values of N that make P return 1, so we can determine it from the number of values of N for which Q halts. In what follows, we shall refer to these quantities as as p_k and q_k respectively.

Unlike p_k, q_k is always finite. Indeed, an upper bound is easily found. Since $\Omega < 1$, only values of k and N such that $\frac{N}{2^k} < 1$ can possibly be less than Ω and thus make Q halt. Since both k and N take only values from the positive integers we also know that $\frac{N}{2^k} > 0$ and thus for a given k, there are less than 2^k values of N for which Q halts and $q_k \in \{0, 1, \ldots, 2^k - 1\}$.

From the value of q_k, it is quite easy to derive the first k bits of Ω. Firstly, note that q_k is equal to the largest value of N such that $\frac{N}{2^k} < \Omega$—unless there is no such N, in which case it equals 0. Either way, its value can be used to provide a very tight bound on the value of Ω: $\frac{q_k}{2^k} < \Omega \leq \frac{q_k+1}{2^k}$. Since Ω is irrational, we can strengthen this to $\frac{q_k}{2^k} < \Omega < \frac{q_k+1}{2^k}$, which means that the first k bits of $\frac{q_k}{2^k}$ are exactly the first k bits of Ω.

This gives some nice results connecting q_k and Ω. The first k bits of $\frac{q_k}{2^k}$ are just the bits of q_k when written with enough leading zeros to make k digits in total. Thus q_k, when written in this manner, provides the first k bits of Ω. Additionally, we can see that q_k is odd if and only if the k-th bit of Ω is 1.

Now that we know the power and flexibility of q_k, it is a simple matter to follow Chaitin in bringing these results to number theory. The function computed by Q is r.e. so, by the DPRM Theorem, there must be a family of Diophantine equations

$$\chi_2(k, N, x_1, \ldots, x_m) = 0 \tag{10.10}$$

that has a solution for specified values of k and N if and only if Q halts when given these values as inputs. Therefore, for a particular value of k, this equation only has solutions for values of N between 0 and $2^k - 1$ with the number of solutions, q_k, being odd if and only if the k-th bit of Ω is 1.

This new family of Diophantine equations improves upon the original one in a couple of ways. Whereas the first method expressed the bits of Ω in the fluctuations between a finite and infinite amount of values of N that

give solutions, the second keeps this value finite and bounded, with the bits of Ω expressed through the more mundane property of parity. It is the fact that this quantity is always finite that leads to many of the new features of this family of Diophantine equations. p_k is infinite when the k-th bit of Ω is 1 and, since there is only one way in which it can be infinite, it can provide no more than this one bit of information. On the other hand, q_k can be odd (or even) in 2^{k-1} ways, which is enough to give $k-1$ additional bits of information, allowing the first k bits of Ω to be determined.

The fact that q_k is always finite also provides a direct reduction of the problem of determining the bits of Ω to Hilbert's Tenth Problem. To find the first k bits of Ω, one need only determine for how many values of N the new family of Diophantine equations has solutions. Since we know that there can be no solutions for values of N greater than or equal to 2^k, we could determine the first k bits of Ω from the solutions to 2^k instances of Hilbert's Tenth Problem. In fact, we can lower this number by taking advantage of the fact that if there is a solution for a given value of N then there are solutions for all lower values. All we need is to find the highest value of N for which there is a solution and we can do this with a bisection search, requiring the solution of only k instances of Hilbert's Tenth Problem.

Finally, the fact that q_k is always finite allows the generalisation of these results from binary to any other base, b. If we replace all above references to 2^k with b^k we get a new program, Q_b, with its associated family of Diophantine equations. For this family, the value of q_k now gives us the first k digits of the base b expansion of Ω: it is simply the base b representation of q_k with enough leading zeroes to give k digits. The value of the k-th digit of Ω is simply q_k mod b.

Chaitin [3] did not stop with his Diophantine representation of Ω, but instead moved to exponential Diophantine equations where his result could be presented more clearly. He made this move to take advantage of the theorem that all r.e. sets have *singlefold* exponential Diophantine representations, where a representation is singlefold if each equation in the family has at most one solution.

We can denote the singlefold family of exponential Diophantine equations for the program P by

$$\chi_1^e(k, N, x_1, \ldots, x_m, 2^{x_1}, \ldots, 2^{x_m}) = 0 \tag{10.11}$$

For a given k, this equation will have exactly one solution for each of in-

finitely many values of N if the k-th bit of Ω is 1 and exactly one solution for each of finitely many values of N if the k-th bit of Ω is 0. We can make use of this to express the bits of Ω through a more intuitive property.

If we treat N in this equation as an unknown instead of a parameter, we get a new (very similar) family of exponential Diophantine equations with only one parameter

$$\chi_1^e(k, x_0, x_1, \ldots, x_m, 2^{x_1}, \ldots, 2^{x_m}) = 0 \tag{10.12}$$

Since the previous family was singlefold and N has become another unknown, there will be exactly one solution to this single parameter family for each value of N that gave a solution to the double parameter family. Thus, (10.12) has infinitely many solutions if and only if the k-th bit of Ω is 1.

This same approach can be used with our method [9]. There is a two-parameter singlefold family of exponential Diophantine equations for Q and this can be converted to a single parameter family of exponential Diophantine equations

$$\chi_2^e(k, x_0, x_1, \ldots, x_m, 2^{x_1}, \ldots, 2^{x_m}) = 0 \tag{10.13}$$

with between 0 and $2^k - 1$ solutions, the quantity being odd if and only if the k-th bit of Ω is 1.

Finally, we have also shown [9] that both Chaitin's finitude-based method and our parity-based method can be used to generate polynomials for Ω. For a given family of Diophantine equations with two parameters,

$$D(k, N, x_1, \ldots, x_m) = 0 \tag{10.14}$$

we can construct a polynomial, W, where

$$W(k, x_0, x_1, \ldots, x_m) \equiv x_0 \left(1 - (D(k, x_0, x_1, \ldots, x_m))^2\right). \tag{10.15}$$

Note that the parameter, N, is again treated as an unknown and thus denoted x_0.

If we restrict the values of the variables to positive integers then, for a given k, this polynomial takes on exactly the set of all values of N for which (10.14) has solutions. We can thus use this method on $\chi_1 = 0$ and $\chi_2 = 0$, generating polynomials that express p_k and q_k in the number of distinct positive integer values they take on for different values of k. We therefore have a polynomial whose number of distinct positive integer values fluctuates from odd to even and back in an algorithmically random manner as a

parameter k is increased.

Our result has now been further extended by Matiyasevich [8].

There are thus many ways in which algorithmic randomness is manifested in number theory. While finding whether solutions exists for certain equations is undecidable, finding the quantity of solutions or even just whether this is finite or infinite, odd or even, is much harder. In the long run, even the best computer program can do no better than chance.

References

[1] C.S. Calude. A characterization of c.e. random reals. *Theoretical Computer Science*, 271:3–14, 2002.

[2] G.J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22(3):329–340, July 1975.

[3] G.J. Chaitin. *Algorithmic Information Theory.* Cambridge University Press, Cambridge, 1987.

[4] G.J. Chaitin. *Meta Math! The Quest for* Ω. Pantheon Books, New York, 2005.

[5] Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential Diophantine equations. *Annals of Mathematics, Second Series*, 74(3):425–436, 1961.

[6] David Hilbert. Mathematical problems. lecture delivered before the International Congress of Mathematicians at Paris in 1990. *Bulletin of the American Mathematical Society*, 8:437–479, 1902.

[7] Yuri V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge, 1993.

[8] Yuri V. Matiyasevich. Hilbert's tenth problem and paradigms of computation. In S.B. Cooper, B. Löwe, and L. Torenvliet, editors, *New Computational Paradigms, Proceedings of the First Conference on Computability in Europre*, pages 310–321, Berlin Heidelberg, 2005. Springer-Verlag.

[9] T. Ord and T.D. Kieu. On the existence of a new family of Diophantine equations for Ω. *Fundmenta Informaticae*, 56:273–84, 2003.

Chapter 11

The Secret Number. An Exposition of Chaitin's Theory

Grzegorz Rozenberg[α] and Arto Salomaa[β1]

[α] *Leiden University, Niels Bohrweg 1*
2333 CA Leiden, The Netherlands; rozenber@liacs.nl
[β] *Turku Centre for Computer Science*
Joukahaisenkatu 3–5 B, 20520 Turku, Finland; asalomaa@utu.fi

The paper deals with information-theoretic complexity based on Chaitin's formal notion of a self-delimiting computer. The technical apparatus required for the presentation of the remarkable properties of the halting probability Ω is developed in detail. Special emphasis is on consequences concerning Diophantine problems.

11.1. Introduction

The purpose of this paper is to discuss formally some key issues dealing with randomness, compressibility and undecidability. Our discussions lead also to a comprehensive exposition of the properties of *Chaitin's "secret number"* Ω and the remarkable connection with the undecidability of Diophantine problems. Many of the subsequent arguments follow our exposition in [7].

Our presentation will be largely self-contained. The reader is assumed to be familiar with the basics of formal languages, Turing machines and recursively enumerable sets. For instance, [8] may be consulted in this respect.

A brief description about the contents of the paper follows. The main result about exponential Diophantine representations will be given below in this Introduction. Section 2 contains an informal discussion about randomness versus the compressibility of information. The informal discussion will continue in Section 10. Section 3 deals with Kraft's inequality, prefix-freeness and frequency indicators. The formal notion of a (Chaitin)

[1]Corresponding author

computer is presented in Section 4, where also the importance of being self-delimited will become apparent. Section 5 shows the existence of universal computers and, accordingly, the notion of (program-size) complexity is defined in the following Section. It is independent of the choice of the universal computer. Section 7 proves fundamental inequalities about program-size complexity. Section 8 compares computational and information-theoretic complexity and also discusses Kolmogorov complexity, where no prefix-freeness is required. Section 9 is about the construction of self-delimiting programs. Section 11 is about the formal connection between probability and complexity. It leads to the degree of randomness, Section 12. The formal apparatus suffices to present in Section 13 the remarkable properties of Ω, the halting probability of the universal computer. Resulting conclusions about formal theories and Diophantine problems are presented in Section 14. The paper ends with some general remarks in Section 15.

We will end this Introduction with some well-known undecidability considerations needed for the main results in Section 14. A very good illustration about how "everyday considerations" in mathematics may lead to undecidability is *Hilbert's Tenth Problem*. The problem, proposed by Hilbert in a list of problems at the International Congress of Mathematicians in 1900, is to give an algorithm that will tell whether or not a polynomial equation with integer coefficients has a solution in integers. Nothing was known about undecidability in 1900 and, in view of Hilbert's later attempts to find general decision methods in logic, it remains questionable whether he in 1900 really had in mind the possibility of a negative solution for the Tenth Problem.

Thus, we consider equations of the form

$$(A) \qquad P(x_1, \ldots, x_n) = 0\,,$$

where P is a polynomial with integer coefficients in the variables $x_1, \ldots, x_n$. A simple number-theoretic argument shows that, from the point of view of decidability, it is irrelevant whether we are looking for integer or nonnegative integer solutions for (A). (See, for instance, [7].)

Things become somewhat simpler and sufficient for our purposes if exponential terms are allowed in (A), that is, if exponential Diophantine representations are considered. This will be our approach.

We now give the formal details. It will be convenient for us to look for solutions in nonnegative integers, rather than in arbitrary integers.

Definition 11.1. A set S of ordered n-tuples $(a_1, \ldots, a_n)$, $n \geq 1$, of nonnegative integers is termed *Diophantine* if and only if there are polynomials

$$P(a_1, \ldots, a_n, x_1, \ldots, x_m),\ Q(a_1, \ldots, a_n, x_1, \ldots, x_m)$$

with nonnegative integer coefficients such that we have, for all nonnegative integers $a_1, \ldots, a_n$,

$$(a_1, \ldots, a_n) \in S \text{ if and only if } (\exists x_1 \geq 0) \ldots (\exists x_m \geq 0)$$

$$[P(a_1, \ldots, a_n, x_1, \ldots, x_m) = Q(a_1, \ldots, a_n, x_1, \ldots, x_m)] .$$

Similarly, S is termed *exponential Diophantine*, briefly ED, if and only if exponentiation is allowed in the formation of P and Q, that is, P and Q are functions built up from $a_1, \ldots, a_n, x_1, \ldots, x_m$ and nonnegative integer constants by the operations of addition $A + B$, multiplication AB and exponentiation A^B.

Some additional remarks are in order. For $n > 1$ we speak of Diophantine and exponential Diophantine *relations*. The terms *unary* and *singlefold* are used in connection with exponential Diophantine representations. Here *singlefold* means that, for any n-tuple $(a_1, \ldots, a_n)$, there is at most one m-tuple $(x_1, \ldots, x_m)$ satisfying the equation. *Unary* means that only one-place exponentials, such as 2^x, are used rather than two-place exponentials y^x. One of the easy observations due to number theory, [7], is that it is no loss of generality to restrict the attention to unary representations.

In decidability theory one begins with an interesting set or relation, such as the set of primes or the relation "x is the greatest common divisor of y and z", and looks for a Diophantine or an exponential Diophantine equation representing it, in the sense of Definition 11.1.

Diophantine and exponential Diophantine sets and relations are recursively enumerable. Also the converses of these statements hold true. The converse of the latter statement, often referred to as the *Davis - Putnam - Robinson Theorem* can be stated as follows.

Theorem 11.2. *Every recursively enumerable set and relation possesses a singlefold exponential Diophantine representation.*

The proof of Theorem 11.2 given in [7] is based on a sequence of constructions, each one showing that a set or relation is exponential Diophantine. The exponential Diophantine characterization is obtained via *register*

machines and shows that the representation obtained is singlefold. This will be essential in our final considerations in Section 14.

It is not known whether or not every recursively enumerable set possesses a singlefold Diophantine representation, although it always possesses a Diophantine representation, by the famous Theorem of Matijasevitch. These matters are discussed in detail in [7].

11.2. Compressibility of information and randomness. An informal discussion

How much can a given piece of information be compressed? This is a matter of fundamental scientific importance and also very relevant for undecidability considerations. In this paper we consider the fundamental problem of compressibility of information. Our considerations lead to a mysterious number Ω ("the secret number", "the magic number", "the number of wisdom", "the number that can be known of but not known", to mention only a few possible descriptions) that encodes very compactly any "cornerstones" of undecidability.

The number Ω is between 0 and 1. It encodes, for instance, the Post Correspondence Problem in the following sense. Suppose we are given an instance PCP. Then, if we know a suitable initial segment of the binary representation of Ω, we are able to compute whether or not PCP possesses a solution. If we restrict our attention to instances of a reasonable size, the knowledge of the first 10,000 bits of Ω is already more than enough. Roughly, if we know the first 10,000 bits of Ω, we are able to solve the halting problems of Turing machines describable in less than 10,000 bits. This surely includes the Turing machines looking for counterexamples to the most famous conjectures in mathematics, such as the Goldbach Conjecture and Riemann's hypothesis.

Even a step further can be taken. Consider a formal system F of mathematics with axioms and rules of inference. For any well-formed formula α, design a Turing machine $TM(F, \alpha)$ that checks through all the proofs of F and halts if it finds one for α. A similar Turing machine $TM(F, \sim \alpha)$ is designed for the negation of α. Again, if we know a sufficiently long initial segment of Ω then we can decide whether α is provable, refutable or independent. The required length depends on F and α — in all reasonable cases the first 10,000 bits of Ω will suffice. This surely justifies the

attributes attached above to Ω.

We use bits as fundamental units of information. A bit indicates a choice between two possibilities, and any piece of information can be encoded as a sequence of bits. Certainly in many cases there are redundant bits in the sequence. In other words, the same sequence can be described in some other way that gives rise to a shorter sequence. This is *per se* the case if the bit sequence is the result of encoding some text in a natural language. Natural languages abound redundancies that have been estimated numerically for various languages and various types of text. The existence of redundancies is obvious because noise in the information channel may distort or delete some letters, and still nothing is lost in the piece of information. Also in classical cryptography cryptanalytic attacks are often based on redundancy properties (unbalanced frequency of individual letters, pairs of letters, etc.) of natural languages.

We now investigate the possibility of compressing information, making sequences of bits shorter. Our first goal is to formalize the notion of *compressibility*. The following construction contains some additional aspects to be considered.

Construction 1. Consider the following two sequences of bits, both of length 16:

$$1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0$$
$$0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,1$$

The first follows an obvious pattern: 10 written 8 times. No such uniform pattern is visible in the second sequence. In fact, the second sequence was generated by coin tosses. Tossing the coin 16 times can produce each of the 2^{16} binary sequences of length 16, and each one of them, including the two mentioned above, has exactly the same probability. Still, it is harder to believe of the first than of the second that it results from coin tosses.

The pattern can be used to compress the first sequence. However, even under some favorable notational conventions, "8 times 10" does not necessarily compress it to less than 16 bits. The matter is entirely different if we are dealing with longer sequences of bits. "1048576 times 10" certainly describes the sequence of repetitions more compactly than the sequence of 2097152 bits $1010\ldots10$. Further advantage can be taken of the fact that $1048576 = 2^{20}$. In general, the most compact way of describing a short sequence is just to write down the sequence. There is no clear borderline

to tell which method is preferable for dealing with sequences such as those in Construction 1.

There may be other ways to compress information than detected patterns. There is no pattern visible in tables of trigonometric functions. Even tables of a modest size give rise to a rather long sequence of bits if everything is expressed as a single sequence. However, a much more compact way to convey the same information is to provide instructions for calculating the tables from the underlying trigonometric formulas. Such a description is brief and, moreover, can be used to generate tables of any size.

Usually no such compression method can be devised for tables presenting empirical or historical facts. For instance, there are books presenting the results of the gold medal winners in each event in each of the Olympic Games since 1896. As regards such information, the amount for compression is negligible, especially if attention is restricted to the least significant digits. Since the results tend to improve, there are regularities in the most significant digits, even to the extent that predictions can be made for the next Olympic Games. In general, as regards empirical data, compression can be linked with *inductive inference* and inductive reasoning as it is employed in science: observations presented as sequences of bits are to be explained and new ones are to be predicted by theories. For our purposes in this paper, it is useful to view a theory as a computer program to reproduce the observations actually made. The scientist searches for minimal programs, and then the amount of compression can be measured by comparing the size of the program with the size of the data. This leads also to a plausible definition concerning what it means that the data are *random*: no compression is possible. In other words, the most concise way of representing the data is just to list them.

Randomness will be a central theme in our considerations. We want to mention one important aspect already at this point. There are various statistical tests for disclosing deviations from randomness. We consider here potentially infinite sequences of bits or digits, such as the decimal expansion of π. A sequence is *normal* if and only if each bit or digit, as well as each block of any length of bits or digits, occurs with equal asymptotic frequency. Clearly, if a sequence does not qualify as normal, then it is not intuitively viewed as a random sequence. On the other hand, normality does not guarantee randomness, no matter whether we view randomness intuitively

or in the sense of incompressibility. For instance, *Champernowne's number*

$$0.12345678910111213141516171819202 1\ldots ,$$

obtained by writing all integers in decimal notation, one after the other, is normal. Intuitively it is non-random. For any i, the ith digit is easily calculated from i and, consequently, long initial segments can be compressed arbitrarily because the formula for the ith digit is independent of the length of the segment. A similar compression is possible for the decimal expansion of the number π because the whole expansion can be generated by a fixed program. Although no proof has been given so far, it is generally conjectured that the expansion of π is normal in the sense described above. The observed statistical data also support this conjecture.

The identification of randomness and incompressibility is feasible also because of the following reason. A gambler who knows the rule governing π or Champernowne's number wins an infinite gain against a gambling casino if the latter produces "random" numbers according to one of the two sequences. Clearly, an intuitive precondition for a sequence being random is that no gambling strategy can produce an infinite gain against that sequence. The knowledge of an initial segment in a random sequence, as opposed to complete ignorance of it, gives no advantage for bets concerning the continuation.

The facts presented in Construction 1 can be viewed as an introduction to the subsequent formal discussion. Construction 1 shows that some central issues in science, such as inductive reasoning and randomness, are just different aspects of compressibility. In order to formalize the idea "no program shorter than the sequence itself can produce the sequence", we have to formalize the notion of a program. For this purpose we present an abstract notion of a computer, due to *Chaitin*. (An intuitive picture of the abstract notion will be given below in Construction 2.) First some technical details about prefix-freeness are needed.

11.3. Prefix-freeness and Kraft's inequality

A language L, finite or infinite, is termed *prefix-free* if and only if no word in L is a prefix of another word in L. For instance, the language $\{a^i b | i = 0, 1, 2, \ldots\}$ is prefix-free. The only prefix-free language containing the empty word λ is the language $\{\lambda\}$.

Words in a prefix-free language L can be used to encode letters from another alphabet. For instance, the subset $\{a^ib|0 \leq i \leq 5\}$ of the above language encodes the letters of the alphabet $V_6 = \{a_0, a_1, a_2, a_3, a_4, a_5\}$ in a natural fashion. Moreover, this encoding has the property of unique decodability: a word over the alphabet $\{a, b\}$ can be decoded in at most one way as a word over V_6. For instance, the word bba^2ba^4b can be decoded only as $a_0a_0a_2a_4$, whereas ba can be decoded in no way.

The property of unique decodability is satisfied always when words in a prefix-free language L are used to encode letters of an alphabet V. For assume that the encodings of two different words $a_1 \dots a_i$ and $b_1 \dots b_j$, where the a's and b's are letters, coincide. We may assume that $a_1 \neq b_1$ because, otherwise, we can divide by the encoding of a_1 from the left. But then the encodings of the two words coincide only if the encoding of a_1 is a prefix of the encoding of b_1, or vice versa, which contradicts the prefix-freeness of L. The argument shows also that decoding from left to right is instantaneous: the remainder of the word does not affect the decoding.

The inequality presented in the next theorem is customarily referred to as the *Kraft inequality*. (See [1] for more details.) The inequality is important in our discussions concerning probabilities. Although we mostly consider binary alphabets in the sequel, the theorem is presented for alphabets with $p \geq 2$ letters.

We begin with the following definition.

Definition 11.3. For a word w over the alphabet $V_p = \{a_1, \dots, a_p\}$, we define the *frequency indicator* of w, in symbols $FI(w)$, by

$$FI(w) = p^{-|w|} .$$

For a language L over V_p, we define the frequency indicator $FI(L)$ to be the sum of the frequency indicators of the words in L.

Thus, $FI(L)$ is a rational number if L is finite. If L is infinite, $FI(L)$ is either a real number or ∞ (depending on whether or not the series converges).

Theorem 11.4. *Every prefix-free language L satisfies $FI(L) \leq 1$.*

Proof. If $FI(L) > 1$ holds for an infinite language L, then also $FI(L_1) > 1$ holds for a finite subset L_1 of L. Moreover, every subset of a prefix-free language is prefix-free. Consequently, it suffices to establish

the claim for a given finite language L.

We claim that, for all $i \geq 1$,

$$(*) \qquad FI(L^i) = (FI(L))^i \, .$$

$(*)$ is established inductively, the basis $i = 1$ being obvious. Assuming that $(*)$ holds, consider L^{i+1}. Clearly, L^{i+1} is prefix-free and all of its words can be represented uniquely in the form xy, where $x \in L$ and $y \in L^i$. Assume that $L = \{x_1, \ldots, x_r\}$ and denote

$$FI(x_j) = p^{-|x_j|} = t_j \, , \quad 1 \leq j \leq r \, .$$

Then

$$FI(L^{i+1}) = \sum_{j=1}^{r} t_j FI(L^i) = FI(L^i) \sum_{j=1}^{r} t_j$$

$$= (FI(L))^i FI(L) = (FI(L))^{i+1} \, .$$

Here the inductive hypothesis, the unique product representation mentioned above, as well as the obvious equation $FI(xy) = FI(x)FI(y)$ have been used.

Let m be the length of the shortest and M the length of the longest word in L. (Possibly $m = M$.) Consequently, every word in L^i is of length $\geq mi$ and $\leq Mi$. This means that there are at most $(M-m)i+1$ different lengths possible among the words of L^i. On the other hand, $FI(L_1) \leq 1$ clearly holds for any language L_1 such that all words in L_1 are of the same length. These observations give us the estimate

$$FI(L^i) \leq (M-m)i + 1$$

and hence, by $(*)$,

$$(**) \qquad (FI(L))^i \leq (M-m)i + 1 \, , \quad \text{for all } \; i \geq 1 \, .$$

Since $M - m$ is a constant independent of i, $FI(L) > 1$ would contradict $(**)$ for values of i large enough. This proves Theorem 11.4

The inequality of Theorem 11.4 need not be strict. The language $L = \{a^i b | i = 0, 1, 2, \ldots\}$ considered above satisfies

$$FI(L) = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots = 1 \, .$$

The equation holds also for some finite languages, for instance,

$$FI(\{a,b\}) = \frac{1}{2} + \frac{1}{2} = FI(\{a,ba,bb\}) = \frac{1}{2} + \frac{1}{4} + \frac{1}{4}$$

$$= FI(\{aa,ab,ba,bb\}) = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1\,.$$

All of these languages are maximal with respect to prefix-freeness: if a new word over $\{a,b\}$ is added to the language then the prefix-freeness is lost. This follows either by a direct argument or by Theorem 11.4. Observe also that the converse of Theorem 11.4 fails: the inequality $FI(L) \leq 1$ does not imply that L is prefix-free.

11.4. Computer: a formal notion

We are now ready to introduce a formal notion of a computer very suitable for our purposes.

Consider the binary alphabet $V = \{0,1\}$. (Thus, FI will later be defined for $p = 2$.) For partial functions $f : V^* \times V^* \to V^*$, we consider also "projections" $f_y : V^* \to V^*$, $y \in V^*$, defined by

$$f_y(x) = f(x,y)\,.$$

Definition 11.5. A *computer* is a partial recursive function $C : V^* \times V^* \to V^*$ such, for all $v \in V^*$, the domain of C_v is prefix-free.

Hence, the basic requirement is that whenever $C(u,v)$ is defined (*converges* in customary terminology of recursive functions) and u is a prefix of u' with $u \neq u'$, then $C(u',v)$ is not defined (*diverges*).

Construction 2. We now explain the above definition in terms of a concrete machine model that can be viewed as a modification of the Turing machine. Briefly, u is the *program* and v is the *input*. Here the "program" is understood as a description of the computing strategy in the same sense as a universal Turing machine is given a description of an individual Turing machine, which it is supposed to simulate. Exactly as in case of ordinary Turing machines, the step by step moves of our computer follow a previously given finite table that completely determines the computation for the argument (u,v).

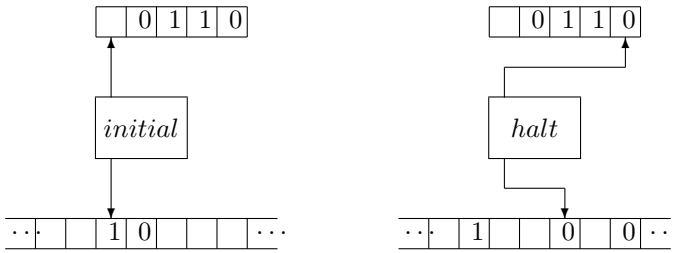
The computer C has two tapes, a *program tape* and a *work tape*. The program tape is finite. Its leftmost square contains a blank, and each of the

remaining squares contains either a 0 or a 1. It is a read-only tape, and the reading head can move only to the right. At the start of the computation, the program u occupies the whole program tape except for the leftmost blank square which is scanned by the reading head.

The work tape is potentially infinite in both directions. Each of its squares contains a blank, 0 or 1. At the beginning all squares are blank except that the input v is written on consecutive squares, and the read-write head scans the leftmost of them. The read-write head can move in both directions. The computer has finitely many internal *states*, among which are a specific *initial* and a specific *halt* state. At the start of the computation, the computer is in the initial state, and no further action is possible in the halt state. (A minor additional detail is that u and/or v may equal the empty word λ.)

The behavior is defined similarly as for ordinary Turing machines. The triple consisting of the current state and the currently scanned symbols from the program and work tapes (there are three possibilities for both of the latter) determine each of the following: (i) the next state, (ii) the symbol written on the work tape (three possibilities), (iii) the move of the reading head (stay or move one square to the right), (iv) the move of the read-write head (stay or move one square to the right or move one square to the left). Thus, each particular computer can be defined by specifying its behavior by a finite table.

The computation of a computer C, started with a program u and input v, is a success if C enters the halt state when the reading head is scanning the rightmost square of the program tape. In this case the output value $C(u, v)$ is read from the work tape, starting from the square scanned by the read-write head and extending to the first blank square. Thus, although called briefly "work tape", the work tape acts also as an *input* and an *output tape*. If the computation is a failure, that is, if the halt configuration described above is not reached, then $C(u, v)$ is not defined. The computation of $C(0110, 10) = 0$ is depicted as follows:



It can now be shown that this concrete machine model computes exactly the partial recursive functions specified in the abstract definition of a computer given before Construction 2. Indeed, the partial recursive functions computed by the machine model must satisfy the additional condition of prefix-freeness. This follows because the machine is allowed neither to run off the right end of the program tape nor to leave some part of the program unread. Thus, $C(u, v)$ and $C(uu_1, v)$, $u_1 \neq \lambda$, can never both be defined. Observe, however, that $C(u, v)$ and $C(uu_1, v')$ can both be defined.

Conversely, we show how a concrete machine C can simulate an abstract computer C'. The basic idea is that C moves the reading head on the program tape only when it is sure that it should do so.

Given a program u and an input v, C first ignores u and starts generating on its work tape, just as an ordinary Turing machine, the recursively enumerable set $X = \{x | C'(x, v) \text{ is defined}\}$. This is done by "dovetailing" through computations for all x. Denote by u_1 the prefix of u already read; initially $u_1 = \lambda$. (C keeps u_1 on its work tape.) All the time C keeps checking whether or not u_1 is a prefix of some element of X already generated. If it finds an x such that $u_1 = x$, C goes to the halt state, after first producing the output $C'(u_1, v)$ on its work tape. If C finds an x such that u_1 is a proper prefix of x, then it reads the next symbol a from the program tape and starts comparisons with $u_1 a$.

This works. If $C'(u, v)$ is defined, C will eventually find u. It then compares step by step the contents of the program tape with u, and halts with the output $C'(u, v)$ if the program tape contains u. If the program tape contains something else, C does not reach a correct halting configuration. This is also the case if $C'(u, v)$ is not defined. Observe that in this case $C'(u_1, v)$ may be defined, for some prefix u_1 of u. Then C finds u_1 and

goes to the halt state but the halting configuration is not the correct one: a portion of the program tape remains unread.

11.5. Universal computer

We now continue to develop the abstract notions.

Definition 11.6. A computer U is *universal* if and only if, for every computer C, there is a *simulation constant* $\mathrm{sim}(C)$ such that, whenever $C(u, v)$ is defined, there is a program u' (that is, a word over the binary alphabet $V = \{0, 1\}$) such that

$$U(u', v) = C(u, v) \text{ and } |u'| \leq |u| + \mathrm{sim}(C) .$$

The following result shows that our definition is not vacuous.

Theorem 11.7. *There is a universal computer.*

Proof. Consider an enumeration of all computers, that is, of all tables defining a computer: C_0, C_1, ... Clearly, the partial function $F : N \times V^* \times V^* \to V^*$ defined by

$$F(i, u, v) = C_i(u, v), \quad i \in N; \quad u, v \in V^* ,$$

is partial recursive, and so is the partial function $U : V^* \times V^* \to V^*$ defined by

$$U(0^i 1u, v) = C_i(u, v) .$$

Moreover, for each v, the domain of the projection U_v is prefix-free. This follows because all projections of each C_i possess the required property of prefix-freeness. Consequently, U is a universal computer with $\mathrm{sim}(C_i) = i + 1$, for each i.

Thinking of the machine model, after reading i 0's from the program tape, U has on its work tape a description of C_i (part from the input v that it has already at the beginning). When U meets the first 1 on the program tape, it starts to simulate the computer C_i whose description it has at that moment on its work tape. (Alternatively, U can store only i on its work tape and, after seeing 1 on the program tape, compute C_i from i.) This concludes the proof of Theorem 11.7.

Universal computers are by no means unique. A different U results, for instance, from a different enumeration of the computers C_i. However, from

now on we consider a *fixed universal computer* U and speak of *the universal computer* U.

Consider the following ordering (alphabetical and according to the length) of the set of words over $V = \{0, 1\}$:

$$\lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots .$$

We use the notation $\min(L)$ for the first word of L. If L is empty, $\min(L)$ is undefined. We now choose one particular program for a word.

Definition 11.8. Given $w \in V^*$, the *canonical program* w^* for w is defined by

$$w^* = \min\{u \in V^* | U(u, \lambda) = w\} .$$

Thus, w^* is the first (and consequently also shortest) program of the universal computer U producing w, when U is started with the empty work tape. The next theorem contains some simple observations about the canonical program.

Theorem 11.9. *The function $f : V^* \to V^*$ defined by $f(w) = w^*$ is total. For any w, $U(w^*, \lambda) = w$ and $w^* \neq \lambda$.*

Proof. Given w, we consider the computer C such that $C(\lambda, \lambda) = w$. Thus, C just prints w on its originally empty work tape, the whole action being embedded in its internal states. (Clearly, we can use also the abstract definition and conclude that a partial recursive function C with the required properties exists. In what follows we use the abstract and concrete notions interchangeably.) By Theorem 11.7, $U(u', \lambda) = w$, for some u'. Hence, the set $\{u \in V^* | U(u, \lambda) = w\}$ is not empty. Since $f(w)$ is defined for an arbitrary w, we conclude that f is total. The equation $U(w^*, \lambda) = w$ is clear by the definition of w^*, and $w^* \neq \lambda$ follows because the domain of U_λ is prefix-free. (Indeed, assume that $U(\lambda, \lambda) = w$. Consider a word $w_1 \neq w$. There is a w' such that $U(w', \lambda) = w_1$. Since $w_1 \neq w$ we must have $w' \neq \lambda$. This implies that the domain of U_λ is not prefix-free.) We have established Theorem 11.9.

11.6. Complexity

We now define the notion of the *complexity* (that could also be called *program-size complexity*) associated to a word $w \in V^*$.

Definition 11.10. The *complexity* of a word w with respect to a computer

C, equals

$$H_C(w) = \min\{|u| \mid u \in V^* \text{ and } C(u, \lambda) = w\} .$$

Again, min is undefined if the set involved is empty. We denote briefly $H(w) = H_U(w)$.

Thus, $H(w)$ is the length of the minimal program for U to compute w when started with the empty work tape. We can view the ratio between $H(w)$ and $|w|$ as the *compressibility* of w. We want to emphasize that, intuitively, $H(w)$ should be understood as the *information-theoretic* (or program-size) complexity of w, as opposed to the computational complexity of w. This is also why we have chosen the notation H, standard in information theory in entropy considerations. $H(w)$ could also be referred to as the *algorithmic entropy*.

We define next the *conditional complexity* of w, given $t \in V^*$.

Definition 11.11. For a computer C, the *conditional complexity* of a word $w \in V^*$ with respect to a word $t \in V^*$ equals

$$H_C(w/t) = \min\{|u| \mid u \in V^* \text{ and } C(u, t^*) = w\} .$$

We denote briefly $H_U(w/t) = H(w/t)$, and speak of the *conditional complexity* of w with respect to t. It is immediate by Theorem 11.9 that $H(w)$ and $H(w/t)$ are defined for all w and t.

The complexities defined depend on the computer C. This is true also as regards $H(w)$ and $H(w/t)$, because they depend on our chosen universal computer. The next theorem asserts that words possess also an inherent complexity, independent of the computer. This holds both for plain and conditional complexity. Although easy to prove, this result is of fundamental importance for the whole theory. The result is often referred to as the *Invariance Theorem*. It says that the universal computer is asymptotically optimal. It does not say much about the complexity of an individual word w, because the constant involved may be huge with respect to the length of w.

Theorem 11.12. *For every computer C, there is a constant A_C such that*

$$H(w) \leq H_C(w) + A_C \text{ and } H(w/t) \leq H_C(w/t) + A_C ,$$

for all w and t.

Proof. The simulation constant $\mathrm{sim}(C)$ satisfies the requirements for A_C and, hence, the theorem follows. Observe that $H_C(w)$ and $H_C(w/t)$ might be undefined. We may either agree that the theorem concerns defined values only, or consider the undefined values as ∞.

Recall that we fixed the universal computer U in a rather arbitrary and certainly in a very nonspecific way. According to Theorem 11.12, this does not matter. For any two universal computers U and U', there is a constant $A_{U,U'}$ such that

$$(*) \qquad |H_U(w) - H_{U'}(w)| \leq A_{U,U'} \ ,$$

for all w. The same result holds for the absolute value of the difference of the conditional complexities as well. Observe, however, that an analogous result does not hold for arbitrary computers C and C'.

The equation $H(w) = |w^*|$ is an immediate consequence of the definition of w^* and Theorem 11.9.

Sometimes it is convenient to use the 0-notation for the *order of magnitude*, defined for functions on real numbers. By definition, $f(x) = 0(g(x))$ if and only if there are positive constants a and x_0 such that

$$|f(x)| \leq a|g(x)|$$

holds for all $x \geq x_0$. Thus, the above inequality $(*)$ can be written as

$$H_U(w) = H_{U'}(w) + 0(1) \ .$$

Let $\varphi(x, y)$ be a *pairing function*, that is, a recursive bijection of $V^* \times V^*$ onto V^*. Such a pairing function is obtained, for instance, from the ordering of V^* using a pairing function for nonnegative integers. We now present the latter in detail. Consider the function $\varphi(i, x)$ defined by the table

	0	1	2	3	4	5	...
0	0	1	3	6	10	15	...
1	2	4	7	11	...		
2	5	8	12	...			
3	9	13	⋮				
4	14	⋮					
⋮	⋮						

Thus, values are plotted to the table in the increasing order of magnitude along the diagonals. The function φ can be defined also by the quadratic expression

$$\varphi(i,x) = \frac{1}{2}(i+x+1)(i+x)+1\,.$$

The function φ establishes a one-to-one correspondence between pairs (i,x) and nonnegative integers. Hence, there are "inverse components" $\varphi_1(x)$ and $\varphi_2(x)$ such that the equation

$$\varphi(\varphi_1(x),\varphi_2(x)) = x$$

holds for all x. These observations are immediate from the table representation of φ. In particular, $\varphi_1(x)$ (resp. $\varphi_2(x)$) is the index of the row (resp. column) in which x lies. Thus,

$$\varphi_1(15) = 0\,,\quad \varphi_2(15) = 5 \ \text{ because } \ \varphi(0,5) = 15\,.$$

The table representation also yields immediately an effective procedure for computing the values $\varphi(i,x)$, $\varphi_1(x)$, $\varphi_2(x)$

Given a computer C, we define

$$H_C(w,t) = H_C(\varphi(w,t))$$

and again $H(w,t) = H_U(w,t)$. Intuitively, $H(w,t)$ is the length of the shortest program that outputs w and t in a way that tells them apart. The next theorem is an exercise concerning symmetry.

Theorem 11.13. $H(w,t) = H(t,w) + 0(1)$.

Proof. The idea is to use the "inverse components" of φ to carry out the commuting. Thus, let $\varphi_i : V^* \to V^*$, $i = 1,2$, be functions such that

$$\varphi(\varphi_1(w),\varphi_2(w)) = w\,,$$

for all w. Consider the computer C, defined by the condition

$$C(x,\lambda) = \varphi(\varphi_2(U(x,\lambda)),\varphi_1(U(x,\lambda)))\,.$$

We now use the fact that, for all w,

$$H(w) \le H_C(w) + \mathrm{sim}(C)\,.$$

Consequently, we compute

$$H(w,t) = H(\varphi(w,t)) \leq H_C(\varphi(w,t)) + \mathrm{sim}(C)$$

$$= \min\{|x| \mid x \in V^* \text{ and } \varphi(\varphi_2(U(x,\lambda)), \varphi_1(U(x,\lambda))) = \varphi(w,t)\} + \mathrm{sim}(C)$$

$$= \min(|x| \mid x \in V^* \text{ and } \varphi_2(U(x,\lambda)) = w \text{ and } \varphi_1(U(x,\lambda)) = t\} + \mathrm{sim}(C)$$

$$= H(\varphi(t,w)) + \mathrm{sim}(C) = H(t,w) + \mathrm{sim}(C)\,.$$

Here also the definition of H_C has been used. Observe also that the mapping $U(x,\lambda) = U_\lambda(x)$ is surjective: the universal computer is capable of producing any word starting with the empty word as its input. Theorem 11.13 follows.

In fact, the above proof shows that if $g : V^* \to V^*$ is a recursive bijection, then

$$H(w) = H(g(w)) + 0(1)\,.$$

To see this, it suffices to consider the computer C defined by the condition

$$C(x,\lambda) = g(U(x,\lambda))\,.$$

The above proof uses the recursive permutation

$$g(x) = \varphi(\varphi_2(x), \varphi_1(x))\,.$$

11.7. Fundamental inequalities

The next theorem is in many ways fundamental in our considerations. The Theorem presents further results concerning the interrelation between $H(w)$, $H(w/t)$ and $H(w,t)$. It consists of several assertions. In (ii) the notation $\overline{H(w)}$ means the word over V whose ordinal number in our ordering of V^* equals $H(w)$.

Theorem 11.14.

(i) $H(w/w) = 0(1)$.
(ii) $H(\overline{H(w)}/w) = 0(1)$.
(iii) $H(w) \leq H(w,t) + 0(1)$.
(iv) $H(w/t) \leq H(w) + 0(1)$.
(v) $H(w,t) \leq H(w) + H(t/w) + 0(1)$.

(vi) $H(w,t) \leq H(w) + H(t) + 0(1)$.

Proof. Consider (i). Since obviously $H(w/w)$ is always nonnegative, it suffices to show that there is a constant A such that $H(w/w) \leq A$, for all w. Define the computer C by the condition

$$C(\lambda, v) = U(v, \lambda), \quad \text{for all } \ v \in V^* \,.$$

Consequently,

$$C(\lambda, w^*) = U(w^*, \lambda) = w$$

and hence,

$$H_C(w/w) = \min\{|u| \mid u \in V^* \ \text{ and } \ C(u, w^*) = w\} = 0 \,.$$

By Theorem 11.12,

$$H(w/w) \leq H_C(w/w) + \mathrm{sim}(C) = \mathrm{sim}(C) = A \,.$$

For (ii), define the computer C by the following condition. Whenever $U(u, \lambda)$ is defined,

$$C(\lambda, u) = \overline{|u|} \,.$$

Since $U(w^*, \lambda) = w$, we obtain

$$C(\lambda, w^*) = \overline{|w^*|} = \overline{H(w)} \,.$$

Consequently,

$$H(\overline{H(w)}/w) \leq H_C(\overline{H(w)}/w) + \mathrm{sim}(C)$$

$$= \min\{|u| \mid u \in V^* \ \text{ and } \ C(u, w^*) = \overline{H(w)}\} + \mathrm{sim}(C)$$

$$= 0 + \mathrm{sim}(C) = \mathrm{sim}(C) \,,$$

from which (ii) follows.

For (iii) and (iv), we use similarly the computers C and C' defined by the conditions

$$C(u, \lambda) = \varphi_1(U(u, \lambda)) \ \text{ and}$$
$$C'(w, t) = U(w, \lambda) \,.$$

Clearly, (vi) is an immediate consequence of (iv) and (v). Hence, it suffices to establish (v). This will be the most interesting construction used in Theorem 11.14.

We *claim* that there is a computer C satisfying the following property. Whenever

$$(*) \qquad U(u, w^*) = t \text{ and } |u| = H(t/w)\,,$$

that is, whenever u is a minimal-size program for U to compute t from w^*, then

$$(**) \qquad C(w^*u, \lambda) = \varphi(w, t)\,.$$

Let us see how (v) follows from this claim. Indeed, by definition, $H_C(w, t)$ is the length of the shortest program for C to compute $\varphi(w, t)$ from the empty word. Since, by $(**)$, w^*u is such a program, we obtain

$$H_C(w, t) \leq |w^*u| = |w^*| + |u| = H(w) + H(t/w)\,.$$

From this (v) follows because always $H(w, t) \leq H_C(w, t) + \mathrm{sim}(C)$.

It remains to verify the claim that there is such a computer C. We follow the abstract definition. Let $C(x, y)$ be the following partial recursive function. For $y \neq \lambda$, $C(x, y)$ is undefined. Let Y be the domain of U_λ, that is,

$$Y = \{u \in V^* | U(u, \lambda) \text{ is defined}\}\,.$$

The following effective procedure is now used to compute the value $C(x, \lambda)$. Elements of the recursively enumerable set Y are generated until, if ever, a prefix v of x is found. Then we write x in the form $x = vu$ and simulate the computations $U(u, v)$ and $U(v, \lambda)$. If both of these computations halt, we output

$$C(x, \lambda) = \varphi(U(v, \lambda), U(u, v))\,.$$

Obviously C is partial recursive. We show that C satisfies the required condition $(**)$, whenever u, w and t satisfy $(*)$. Denote $x = w^*u$, and consider our algorithm for computing $C(x, \lambda)$. Since $U(w^*, \lambda) = w$, we conclude that w^* is in Y and, consequently, will eventually be found as a prefix v of x, yielding the factorization $x = vu = w^*u$. Moreover, it is not possible that another prefix v' of x would be found in this fashion, because then one of the words v and v' would be a prefix of the other, where both of the words v and v' are in Y. However, this would contradict our basic assumption concerning computers: the domain of U_λ is prefix-free.

Once the unique factorization $x = vu$ with $v = w^*$ has been found, both of the computations $U(u, v)$ and $U(v, \lambda)$ halt because

$$U(u, v) = U(u, w^*) = t \text{ and } U(v, \lambda) = U(w^*, \lambda) = w \, .$$

Now we see that the output satisfies

$$C(x, \lambda) = \varphi(U(v, \lambda), U(u, v)) = \varphi(w, t) \, ,$$

as it should according to $(**)$.

We still have to show that $C(x, y)$ is a computer, that is, the domain of C_y is prefix-free, for all y. This is clear for $y \neq \lambda$. Consider the domain of C_λ. Assume that x is a prefix of y and that $C(x, \lambda)$ and $C(y, \lambda)$ are both defined. This implies that we obtain the decompositions

$$x = v_x u_x \ , \ y = v_y u_y$$

and, moreover, each of the values

$$U(u_x, v_x) \, , \ U(v_x, \lambda) \, , \ U(u_y, v_y) \, , \ U(v_y, \lambda)$$

is defined. Since x is a prefix of y, one of the words v_x and v_y is a prefix of the other. But this is possible only if $v_x = v_y$, because the domain of U_λ is prefix-free. We now conclude that u_x is a prefix of u_y. Because both u_x and u_y belong to the prefix-free domain of U_{v_x} $(= U_{v_y})$, we make the further conclusion that $u_x = u_y$ and, consequently, $x = y$. This shows that the domain of C_λ is prefix-free. We have completed the proof of (v) and, consequently, of Theorem 11.14.

We introduce a further notion. The amount by which the conditional complexity $H(w/t)$ is less than the (unconditional) complexity $H(w)$ can be viewed to indicate how much t contains information about w. The notion is again first defined for arbitrary computers and then for the universal computer.

Definition 11.15. The *algorithmic information in t about w* with respect to the computer C equals

$$I_C(t : w) = H_C(w) - H_C(w/t) \, ,$$
$$I(t : w) = I_U(t : w) \, .$$

One would expect that the information contained in w about w is approximately the same as the information contained in w, and that the information contained in λ about w, and vice versa, amounts to nothing. Also the other formal statements presented in the next theorem are plausible from the intuitive point of view.

Theorem 11.16.

(i) $I(t : w) \geq 0(1)$.
(ii) $I(t : w) \leq H(t) + H(w) - H(t, w) + 0(1)$.
(iii) $I(w : w) = H(w) + 0(1)$.
(iv) $I(w : \lambda) = 0(1)$.
(v) $I(\lambda : w) = 0(1)$.

Proof. Assertions (i)–(iii) follow by Theorem 11.14, from assertions (iv), (v) and (i), respectively. To prove (iv), we observe the already established (i) and write according to (ii):

$$\begin{aligned} I(w : \lambda) &\leq H(w) + H(\lambda) - H(w, \lambda) + 0(1) \\ &= H(w) - H(w, \lambda) + 0(1) = 0(1) \, . \end{aligned}$$

Here the last equation is established by defining the computer

$$C(u, \lambda) = \varphi_1(U(u, \lambda))$$

and observing that $H(w) \leq H(w, \lambda) + \mathrm{sim}(C)$. The proof of (v) is similar, and Theorem 11.16 follows.

11.8. Computational and information-theoretic complexity

The notion of complexity discussed above could be referred to as the *Chaitin complexity*; the basic concepts and the notation, as well as the definition of the computer C, are from [cha1]. This notion of complexity falls within the framework of *descriptional* or *information-theoretic* complexity, as opposed to *computational* complexity. In the former, one is interested in the *program size*: only the length of the shortest program for a specific task matters, not how hard the computation according to the program will be. For the latter type of complexity, computational complexity, the *amount of resources* such as time needed in the computation is essential.

Construction 3. Many problems concerning regular languages, although decidable, are known to be hard from the point of view of computational complexity. For instance, it has been shown that for some such problems no polynomial space bound exists. We use this fact to show that sometimes the descriptional complexity is "as low as it can be", whereas the computational complexity is high.

Consider regular expressions over the alphabet $\{a, b\}$. Thus, we use Boolean operations, catenation and star to the "atoms" a, b, ϕ. Let α_i, $i = 0, 1, \ldots$, be an ordering of such regular expressions. For instance, we may order the regular expressions first according to length and then alphabetically to get the order α_i. Let us call an index i *saturated* if and only if the regular language denoted by α_i equals $\{a, b\}^*$, that is, its complement is empty. A real number

$$r = .a_0 a_1 a_2 \ldots$$

in binary notation is now defined by the condition: $a_i = 1$ if and only if i is saturated.

Clearly, $r > 0$ because some indices i are saturated. On the other hand, in our ordering that takes the length into account, the first saturated index appears quite late, because at the beginning only languages with a nonempty complement appear. We consider also the language consisting of all prefixes of the binary expansion of r:

$$L_r = \{w \in \{0,1\}^+ | w = a_0 a_1 \ldots a_i, \text{ for some } i \geq 0\} .$$

It is obvious that there is a constant A such that the descriptional complexity of words in L_r is bounded from above by A, provided the length of the word is given. This follows because the algorithm for computing bits of the expansion of r, that is, for testing the emptiness of certain regular languages can be carried out by a fixed computer. Such a computer can be one of the computers C defined above. Then C starts with an empty program tape and with the index i written in binary notation on its work tape, and halts with the output $a_0 a_1 \ldots a_i$. Since the binary expansion of i contains, roughly, $\log i$ bits we obtain the result $H(w) \leq \log |w| + A$ for all w in L_r. The same result is obtained also if r is the decimal or binary expansion of π. This can be viewed as the lowest possible descriptional complexity: the same algorithm produces arbitrarily long prefixes of an infinite sequence. This is a property shared by all computable (recursive) sequences. The estimate $H(w) \leq \log |w| + A$ can be further improved by replacing $|w| = i + 1$ with its descriptional complexity. This yields the

estimate $H(w) \leq H(i+1) + A$.

On the other hand, the computational complexity of r (that can be viewed as the computational complexity of the membership problem in L_r) is very high. Indeed, in order to decide the membership of a word of length i, one has to settle the emptiness of the complement for all of the i regular languages involved. Although both r and π are of essentially the same low descriptional complexity, the computational complexity of r is essentially higher than that of π.

It is natural to ask whether there are reverse examples, that is, cases where the computational complexity is low but the descriptional complexity is high. This is very much a matter of how the setup is defined. It is difficult to find examples as obvious as the ones above.

The descriptional complexity of a word, or the amount of information in a word, can be identified as the size of the smallest program that produces the word when started with a blank memory. In general, according to different models, the amount of information is invariant up to a minor additive term. Descriptional complexity is nowadays usually referred to as the *Kolmogorov complexity*. With the early developments from 1960's in mind, *Solomonoff - Chaitin - Kolmogorov complexity* is perhaps the most appropriate term for descriptional complexity. (See [6] for a historical account concerning these matters.) Our notion of Chaitin complexity could also be called *self- delimiting Kolmogorov complexity*. We now discuss this difference in more detail, and explain also why Chaitin complexity is more appropriate for our purposes.

In general, the *Kolmogorov complexity* of a word w with respect to a description method M, in symbols $K_M(w)$, is the length of the shortest program p such that M with the program p produces w. To get a formal definition, we identify "description methods" with partial recursive functions.

Consider again the binary alphabet $V = \{0, 1\}$, and let $f : V^* \times V^* \to V^*$ be partial recursive.

Definition 11.17. The *Kolmogorov complexity* of a word w in V^* with respect to f is defined by

$$K_f(w) = \min\{|p| \mid f(p, \lambda) = w\} .$$

(The min-operator is undefined when applied to the empty set.) Similarly,

the *conditional Kolmogorov complexity* of w, given t, is defined by

$$K_f(w/t) = \min\{|p| \mid f(p,t) = w\} .$$

Let us immediately compare this definition with our previous definition of (Chaitin) complexity and conditional complexity. The essential difference is that the definition of the Kolmogorov complexity does not require the condition concerning prefix-freeness: it is not required that the domain of

$$f_t(p) = f(p,t)$$

is prefix-free, for all t. We will comment this further in the sequel. Another difference is that t rather than t^* appears as the second argument place of f in the definition of the conditional Kolmogorov complexity. (One consequence of this difference is that $K(w/\lambda) = K(w)$, whereas in general $H(w/\lambda) \neq H(w)$.)

The *Invariance Theorem* holds for the Kolmogorov complexity as well, the proof being essentially the same as for Theorem 11.12. Therefore, the proof of it is omitted.

Theorem 11.18. *There is a partial recursive function f_U (a "universal" function) with the following property. For every partial recursive function f, there is a constant A_f such that*

$$K_{f_U}(w) \leq K_f(w) + A_f \quad \text{and} \quad K_{f_U}(w/t) \leq K_f(w/t) + A_f ,$$

for all w and t.

Similarly as before, we fix a partial recursive function f_U satisfying Theorem 11.18 and use the simple notation K instead of K_{f_U}. We can also define similarly as before

$$K(w,t) = K(\varphi(w,t)) ,$$

where φ is a pairing function. But a remarkable difference is that now we do *not* obtain the estimate $K(w,t) \leq K(w) + K(t) + 0(1)$, analogous to Theorem 11.14 (vi). Let us look into this in more detail.

The construction for Theorem 11.14 (v), used to obtain Theorem 11.14 (vi), does not work for K instead of H. If arbitrary partial recursive functions rather than computers (satisfying the prefix-freeness condition) are used, then the decomposition $x = vu$ is not clear a priori but additional information is needed to tell v and u apart. This amounts to information concerning the length of either w or t. The length of w being logarithmic

in terms of w viewed as a binary number, the estimate corresponding to Theorem 11.14 (vi) reads

$$K(w,t) \leq K(w) + K(t) + 0\,(\log(\min(K(w), K(t)))) \ .$$

It can be shown that the logarithmic fudge term is necessary.

We already pointed out that the Chaitin complexity could be called *self-delimiting Kolmogorov complexity*. Here "self-delimiting" means that the total length of a program must be given within the program itself. In our machine model (the computer C) this is taken care of by the requirement that the computer never runs off the end of the program tape or ignores a part of the program. This leads to prefix-freeness: no program for a successful computation is a prefix of another.

11.9. Self-delimiting programs

Programs are self-delimiting if constructs are provided for beginning and ending a program. This is easily accomplished if end markers are available. The situation is trickier if the whole program is, as in our case, just a sequence of bits. The program might actually describe a formal parametrized procedure, as well as list the values of the parameters. One has to be able to tell apart all of these items in the single bit sequence that constitutes the program. How this requirement (of self-limitedness) can be realized will be illustrated in the next construction.

Construction 4. Given a word w over the binary alphabet, we define the *self-delimiting presentation* $SD(w)$ of w. The idea is to write the length of w written in binary notation in front of w, that is, to consider the word $|w|w$, where $|w|$ is given in binary notation. The following additional trick makes it possible to recognize immediately in a (possibly long) binary sequence, where the length indicator ends and the "proper word" begins. (Although every word over V can be represented in at most one way as the catenation of $|w|$ and w, for some w, an unbounded amount of lookahead is needed to tell the border between $|w|$ and w without the additional trick.) In $|w|$ we write a 0 after every bit except that after the last bit we write a 1. In this fashion we obtain the word $ML(w)$, the "modified length" of w. Finally, $SD(w) = ML(w)w$. Some illustrations are given in the following table.

w	$\|w\|$	$ML(w)$	$SD(w)$
01	10	1001	100101
101001	110	101001	101001101001
1^{15}	1111	10101011	1010101^{17}

Consider now an arbitrary word x over $\{0,1\}$. To present x in the form $x = ML(w)w$, provided this is at all possible, it suffices to find the first bit 1 in x that occurs in an even-numbered position, counted from left to right.

The longer the word w is, the less is the contribution of $ML(w)$ to the length of $SD(w)$. Asymptotically, we have

$$|SD(w)| = |w| + 2\log|w| \,.$$

Thus, the length increases only by an additive logarithmic term in the transition from a word to its self-delimiting presentation. Such a difference by a logarithmic term is often present when comparisons are made between the Kolmogorov complexity and the self-delimiting Kolmogorov complexity (Chaitin complexity). A typical example is Theorem 11.14 (vi). Because K is not self-delimited, a logarithmic term is needed, but it is not needed for H because it is already taken care of in the definitions.

To summarize, one can distinguish in the study of descriptional complexity the self-delimiting version (Chaitin complexity) and the non-self-delimiting version (Kolmogorov complexity). As regards conditional complexity, there is a further possibility for a different definition, depending on whether a word t itself, or the shortest program t^* for it is considered. (We made the latter choice in our definition of complexity.) Different variations in the basic definitions result in somewhat different theories but it is beyond the scope of this contribution to investigate this matter in detail.

For our purposes, the given definition of complexity, using computers C and the resulting prefix-freeness, is the most suitable. Our main purpose is the discussion concerning the secret number Ω, a very compact way of encoding the halting problem (or any of the other undecidable problems, [5]). In this discussion, probabilities play a central role: we will use the Kraft inequality (Theorem 11.4). The self-delimiting version satisfies the essential requirement for prefix-freeness.

11.10. Randomness: intuitive considerations

We now return to the discussion, started already in Section 2, concerning randomness viewed as incompressibility. Before giving the formal definitions, we will begin with some informal considerations. As before, we will talk about words w over the binary alphabet $V = \{0, 1\}$.

A word w is incompressible or random if and only if the shortest program describing w is roughly of the same length as w. An infinite sequence of bits is random if and only if the condition mentioned holds for all prefixes of the sequence. For (finite) words w, randomness is a matter of degree. Depending on the model, a certain additional term to $|w|$ is needed, and the degree of randomness indicates how close the length of the shortest program for w is to the maximal value. "How random is w?" is the proper question to ask in this case because the additional term may be huge or negligible, depending on the length of w. For infinite sequences, there is a sharp distinction between randomness and nonrandomness, because the additional term will eventually become negligible.

Consider the degree of randomness of a word w with $|w| = i > 20$. Call a word w "fairly random" iff the shortest program for describing w is of length $\geq i - 20$. (We have here an arbitrary model in mind, not necessarily the model with computers C and complexity H.) The following argument shows that almost all words are fairly random. Consider a fixed i. Assume that every word over V of length $< i - 20$ is actually a program describing a word of length i. There are

$$2^1 + 2^2 + \cdots + 2^{i-21} = 2^{i-20} - 2$$

nonempty words of length $< i - 20$. Hence, out of the 2^i words of length i, at most $2^{i-20} - 2$ can be described by a program of length $< i - 20$. This holds for an arbitrary i. The ratio between $2^{i-20} - 2$ and 2^i is less than 10^{-6}. Consequently, less than one in a million among the numbers of any given length is not fairly random. If randomness is taken to mean information-theoretic incompressibility in the sense explained above, almost all numbers are fairly random. A formal counterpart of this result will be presented in Theorem 11.23. If a word w over $\{0, 1\}$ is generated by fair coin tosses, the probability is greater than .999999 that the result will be random to the extent indicated in the definition of fair randomness. This would seem to suggest that it is easy to exhibit a specimen of a long fairly random word. One of the conclusions made in the sequel will be that it is actually impossible to do so.

Let us elaborate the latter claim. The claim is that, although most words are random, we will never be able to explicitly exhibit a long word that is demonstrably random. The reason is that the axioms and rules of inference in any formal system T can be described in a certain number, say n, of bits. The system T cannot be used to prove the randomness of any word w much longer than n because, otherwise, a contradiction would arise. If $|w| = m$ is much larger than n and the randomness of w could be proven within T, we could construct a computer C to check through the proofs of T, until the correct one is found. The description of C takes roughly n bits and, thus, we get a program much shorter than m describing w, a contradiction. To put it very simply, we cannot construct a program p to print a word w with $|w| > |p|$ unprintable by programs q with $|q| < |w|$. Chaitin has expressed the matter by saying that if one has ten pounds of axioms and a twenty-pound theorem, then the theorem cannot be derived from the axioms.

Thus, we cannot know a random number but we can still know *of* a random number. In particular, we can know of the secret number Ω: it is the probability that the universal computer U halts when it is started with an empty work tape. (Thus U is started with a pair (u, λ), where the program u is arbitrary.) Before going into formal details, we still present another possibility to encode the halting problem.

Construction 5. We consider now ordinary Turing machines because self-delimitedness is not important here. Let TM_0, TM_1, TM_2, ... be a numbering of all Turing machines, and define a number A by its binary expansion

$$A = .a_0a_1a_2\ldots ,$$

where, for all i, $a_i = 1$ if and only if TM_i halts with the empty tape. We already pointed out in Construction 3 above that computable sequences are never random. However, A is clearly noncomputable and, thus, could be random as far as this matter is concerned. But A is not random. A gambler is able to make an infinite profit by using some infinite subclass of Turing machines with a decidable halting problem. A formal argument concerning the compressibility of A can be based on the following observation. Consider any prefix of A of length n. Suppose we know the *number* m of 1's in this prefix. Then we know also the prefix itself, because we can dovetail the computations of the first n Turing machines, until have found m of them

that have halted. Eventually, this will happen. Thus, information about the prefix of length n can be compressed to information about n and m.

11.11. Probability and complexity

We first present the formal definition of Ω. We consider the binary alphabet $V = \{0, 1\}$. We use the definitions of a computer C and a universal computer U given above. Moreover, exactly as was done above, a fixed universal computer U will be considered all the time. Also the "optimal program" t^* for t is defined as before.

We are now ready for the fundamental definition.

Definition 11.19. *the probability* of a word $w \in V^*$ with respect to a computer C equals

$$P_C(w) = \sum_{\substack{u \in V^* \\ C(u,\lambda)=w}} 2^{-|u|} .$$

The *conditional probability* of w with respect to a word $t \in V^*$ and computer C equals

$$P_C(w/t) = \sum_{\substack{u \in V^* \\ C(u,t^*)=w}} 2^{-|u|}.$$

$P_C(w)$, resp. $P_C(w/t)$, is defined to be 0 if no u as required on the right side exists. The *probability* of w and the *conditional probability* of w with respect to t are defined by

$$P(w) = P_U(w) \text{ and } P(w/t) = P_U(w/t) .$$

Finally, the *halting probability of the universal computer is defined by*

$$\Omega = \sum_{\substack{u \in V^* \\ U(u,\lambda) \text{ converges}}} 2^{-|u|} .$$

In this definition, instead of probabilities, we could speak also of *algorithmic probabilities* or *information-theoretic probabilities*. The justification for the terminology and the interconnection with the classical probability theory will be discussed below. We prove first some formal results.

Theorem 11.20. *The following inequalities hold for all words w and t over V and for all computers C.*

(i) $0 \leq P_C(w) \leq 1$,
(ii) $0 \leq P_C(w/t) \leq 1$,
(iii) $0 \leq \sum_{x \in V^*} P_C(x) \leq 1$,
(iv) $0 \leq \sum_{x \in V^*} P_C(x/t) \leq 1$.

Proof. Recall the definition of the frequency indicator, FI, Definition 11.3. Clearly,

$$P_C(w) = FI(L), \quad \text{where} \quad L = \{u \in V^* | C(u, \lambda) = w\} .$$

By the definition of a computer, the domain L' of C_λ is prefix-free. L is a subset of this prefix-free language and, hence, L itself is prefix-free. The inequality $P_C(w) \leq 1$ now follows by Theorem 11.4.

Observe next that $\sum_{x \in V^*} P_C(x) = FI(L')$, and hence the upper bound in (iii) follows by Theorem 11.4. To get the upper bounds in (ii) and (iv), we use in the same way the domain of C_{t^*} (that is prefix-free by the definition of a computer) and its subset determined by w. The lower bounds in (i)–(iv) are obvious because all terms in the series are nonnegative. Hence, Theorem 11.20 follows.

The next theorem presents an interconnection between probability and complexity.

Theorem 11.21. *For all w, t and C,*

$$P_C(w) \geq 2^{-H_C(w)} \quad \text{and} \quad P_C(w/t) \geq 2^{-H_C(w/t)} .$$

Proof. According to the definition, $P_C(w)$ (resp. $P_C(w/t)$) is a sum of terms, one of which is $2^{-H_C(w)}$ (resp. $2^{-H_C(w/t)}$). The inequalities are formally valid also for undefined values of H_C if we agree that the value is ∞ in this case. This completes the proof.

The next theorem shows that the probabilities lie in the proper interval.

Theorem 11.22. *For all w and t,*

$$0 < P(w) < 1 \quad \text{and} \quad 0 < P(w/t) < 1 .$$

Proof. The inequalities $0 < P(w)$ and $0 < P(w/t)$ follow by Theorem 11.21 because $H(w)$ and $H(w/t)$ are always defined (see Theorem 11.9). By Theorem 11.20, (iii),

$$\sum_{x \in V^*} P(x) \leq 1 \,,$$

and each term in the sum is greater than 0, we must have $P(w) < 1$. The inequality $P(w/t) < 1$ follows similarly by Theorem 11.20, (iv). Consequently, Theorem 11.22 follows.

11.12. Degree of randomness

We begin this section by establishing some upper bounds concerning the cardinalities of certain sets defined in terms of H and P. Intuitively, if randomness is understood as information-theoretic incompressibility, then almost everything is fairly random.

Theorem 11.23. *For all computers C, words t and integers $m, n \geq 1$, we have*

(i) $card\{w \in V^* | H_C(w) < m\} < 2^m$,
(ii) $card\{w \in V^* | H_C(w/t) < m\} < 2^m$,
(iii) $card\{w \in V^* | P_C(w) > \frac{m}{n}\} < \frac{n}{m}$,
(iv) $card\{w \in V^* | P_C(w/t) > \frac{m}{n}\} < \frac{n}{m}$.

Proof. $H_C(w)$ is the length of the shortest u, if any, such that $C(u, \lambda) = w$. Each u gives rise to at most one w, and there are no more than $2^m - 1$ possible u's, since this is the total number of words shorter than m. Hence, (i) follows. The proof for (ii) is similar. Arguing indirectly, we see that if (iii) does not hold, then

$$1 = \frac{n}{m} \cdot \frac{m}{n} \leq \text{card}\{w \in V^* | P_C(w) > \frac{m}{n}\} \cdot \frac{m}{n}$$

$$< \sum_{w \in V^*} P_C(w) \leq 1 \,.$$

Here the last inequality follows by Theorem 11.20, (iii), and the strict inequality holds because we have strictly increased every element in a sum and possibly added new elements. The contradiction $1 < 1$ shows that (iii) holds. Again, the proof for (iv) is similar, and Theorem 11.23 follows.

We want to emphasize that in our definitions the term "probability" is only suggestive. No properties of probabilities have been used in the proofs of Theorems 11.20–11.23. However, this suggestive term is very well justified. Intuitively, $P_C(w)$ coincides with the probability for the computer C to produce the output w when started with a program u generated by coin tosses, and an empty work tape. (The length of the program tape is adjusted according to the program u.) An analogous intuitive point of view can be given as regards the other P-notions. A more formal approach would be to introduce a uniform probabilistic structure s on the alphabet V by defining $s(0) = s(1) = \frac{1}{2}$, and to consider the product space, consisting of infinite sequences of letters of V, provided with the product probability. The details of such an approach are beyond the scope of this article.

We now return to the discussion of randomness. We already pointed out that, for words w, there is no sharp distinction between randomness and nonrandomness. However, one can speak of the *degree of randomness* . For a word w of length i, the degree of randomness of w indicates how close $H(w)$ is to the maximal value $i + H(i)$.

As regards infinite sequences, there is a sharp distinction. We can define explicitly what we mean by a random sequence. Although practically all infinite sequences are random, it is (and will be also in the future!) impossible to exhibit one of which we can prove that it is random. We now give the formal definition.

Definition 11.24. For an infinite sequence

$$B = b_1 b_2 b_3 \dots$$

of elements of $\{0, 1\}$, we let $B_i = b_1 \dots b_i$ be the prefix of length i of B, for $i = 1, 2, \dots$. The sequence B is *random* if and only if there is a constant A such that, for all i,

$$H(B_i) > i - A\,.$$

At the beginning of this paper we pointed out that random sequences must possess certain statistical properties such as being normal. Such properties must be present in all reasonably constructed subsequences as well. In Construction 5, for instance, we found a reasonable subsequence consisting of 1's only — hence the whole sequence is not random. One can formulate the notion of randomness in terms of effectively verifiable statistical tests. What is very pleasing is that this definition of randomness yields exactly

the same random sequences as the definition given above, see [Li Vi] and [Cal].

11.13. Magic bits: properties of the secret number

We now discuss the properties of Ω, the halting probability of the universal computer, the "secret number". Taking $C = U$ in Theorem 11.20, (iii), we obtain first $0 \leq \Omega \leq 1$. Theorem 11.22 shows that the first inequality is strict. That also the second inequality is strict is a consequence of the fact that U cannot halt for all programs used in the proof of the Kraft inequality, Theorem 11.4. Thus, we have

$$0 < \Omega < 1 \, .$$

Let

$$\Omega = .b_1 b_2 b_3 \ldots$$

be the binary expansion of Ω. Ambiguities are avoided by choosing the non-terminating expansion whenever two expansions are possible. (We do not want to exclude the possibility of Ω being rational!) Informally, we refer to the bits b_i, $i \geq 1$, in the expansion of Ω as *magic bits*. Thus, Ω is a real number. We consider also the infinite sequence

$$B = b_1 b_2 b_2 \ldots$$

of letters of $\{0, 1\}$, as well as its prefix of length i,

$$B_i = b_1 \ldots b_i \, , \quad i \geq 1 \, .$$

For all $i \geq 1$, we consider the rational number

$$\Omega_i = .b_1 \ldots b_i \, .$$

We are now in the position to establish the remarkable properties of the number Ω hinted at earlier in this paper. We prove first a result showing that Ω encodes the halting problem in a very compact form. We then establish that B (as defined above from Ω) is random, and proceed to consider the implications to formal axiomatic theories. Any theory is capable of yielding only finitely many bits of B. Thus, we can never know Ω in the sense that we could somehow produce infinitely many bits of B. But we can know *of* Ω in the sense that we can define it formally. Finally, we show that no formal axiomatic theory can ever tell whether a certain Diophantine equation with a parameter has infinitely many solutions. This can possibly

be determined for finitely many parameter values but never for infinitely many, let alone all, of them.

The domain of U_λ, that is, the set

$$DOM(U_\lambda) = \{u \in V^* | U(u, \lambda) \text{ is defined } \}$$

is recursively enumerable. We consider some fixed enumeration of it, where repetitions do not occur. Such an enumeration is obtained by a method customarily referred to as "dovetailing". We have already hinted at this method before: you order the steps in different computations in one sequence, as in the definition of the pairing function.

Thus, we obtain a total recursive injection g of the set of positive integers into V^*. We define, for $n \geq 1$,

$$\omega_n = \sum_{j=1}^{n} 2^{-|g(j)|} .$$

It is obvious that the sequence ω_n, $n = 1, 2, \ldots$, is monotonically strictly increasing and converges to Ω.

Theorem 11.25. *Whenever $\omega_n > \Omega_i$, then*

$$\Omega_i < \omega_n < \Omega \leq \Omega_i + 2^{-i} .$$

Given any i, if we know the first i bits of Ω, we can decide the halting of any program with length $\leq i$.

Proof. The first sentence is an immediate consequence of the inequality

$$2^{-i} \geq \sum_{j=i+1}^{\infty} b_j 2^{-j} .$$

To prove the second sentence, assume that we know B_i. Hence, we are able to compute Ω_i. We now compute the numbers ω_j until we have found an n such that $\omega_n > \Omega_i$. By the properties of ω_j, this is always possible because we know that such an n eventually comes up.

Let u_1 be a word over V of length $i_1 \leq i$. We claim that $U(u_1, \lambda)$ is defined if and only if u_1 is one of the words $g(1), \ldots, g(n)$. The "if"-part of the claim is clear. To prove the "only if" -part, we assume the contrary: $u_1 = g(m)$, where $m > n$. We obtain a contradiction by the following chain of inequalities

$$\Omega > \omega_m \geq \omega_n + 2^{-i_1} \geq \omega_n + 2^{-i} > \Omega_i + 2^{-i} \geq \Omega .$$

This concludes the proof of Theorem 11.25.

Theorem 11.25 justifies the term "magic bits". The knowledge of a sufficiently long prefix B_i enables us to solve any halting problem. The same applies to Post Correspondence Problems as well, because we can design programs to search for a solution of a given PCP. Similarly, we can design programs looking for counterexamples to famous conjectures in classical mathematics, such as the Goldbach Conjecture or Riemann's hypothesis. We have already indicated before how the knowledge of a sufficiently long sequence B_i of magic bits opens even more general vistas. It enables us to decide whether a well-formed formula is, according to a formal theory, a theorem, a nontheorem or independent.

How many magic bits are actually needed, depends of course on the formal theory and also on the programming of the universal computer U. Certainly our programming of U in the proof of Theorem 11.7 was not very economical in this respect. We started the programs with 0^i1, where i is the index of an individual computer C_i. A much more compact programming for U results if we take the programs of the individual computers C_i as such and use the technique of Construction 4 to make words self-delimiting. Then for any conceivably interesting formal theories the knowledge of 10,000 magic bits, that is B_i with $i = 10,000$, is more than enough.

Thus, if an oracle tells us 10,000 magic bits, we are wise enough to settle all halting problems and PCP's that are of some reasonable size, as well as famous conjectures in classical mathematics. Although we are wise in this sense, we still face a task of an enormous computational complexity when we start calculating the numbers ω_n.

We will now prove that Ω is truly random.

Theorem 11.26. *The sequence B is random.*

Proof. We apply the notation from the proof of Theorem 11.25. We showed that whenever $U(u_1, \lambda)$ is defined and $|u_1| \leq i$, then u_1 is one of the words $g(1), \ldots, g(n)$. This leads to the equation

$$(A) \qquad \{U(g(j), \lambda) | 1 \leq j \leq n \text{ and } |g(j)| \leq i\} = \{w | H(w) \leq i\} .$$

Indeed, every word belonging to the left side belongs to the right side and, conversely, if $H(w) \leq i$ then w has a program of length at most i.

Consider now the partial recursive function f from V^* into V^*, defined as follows. Given $x = x_1 \ldots x_t$, $x_j \in V$, find the smallest m, if any, such

that

$$\omega_m > \sum_{j=1}^{t} x_j 2^{-j} .$$

If such an m is found, $f(x)$ is the first word (in lexicographical order) not belonging to the set

$$\{g(j) | 1 \leq j \leq m\} .$$

Let C be the computer defined by

$$C(x, \lambda) = f(U(x, \lambda)) .$$

We now consider an arbitrary prefix B_i and infer

$$\begin{aligned} & H(f(B_i)) \leq H_C(f(B_i)) + \mathrm{sim}(C) \\ & = \min\{|u| \mid C(u, \lambda) = f(B_i)\} + \mathrm{sim}(C) \\ & = \min\{|u| \mid f(U(u, \lambda)) = f(B_i)\} + \mathrm{sim}(C) \\ & \leq \min\{|u| \mid U(u, \lambda) = B_i\} + \mathrm{sim}(C) \\ & = H(B_i) + \mathrm{sim}(C) . \end{aligned}$$

By the equation (A), $H(f(B_i)) > i$. Hence,

$$i - \mathrm{sim}(C) < H(B_i)$$

for all i. This means that B is random, by Definition 11.24 with $A = \mathrm{sim}(C)$.

It is clear by the discussion after Theorem 11.25 that an upper bound n is inherent in every formal theory, such that no prefix B_i with $i > n$ can be produced according to the theory. Theorem 11.26 shows that such an upper bound concerns also the total number of bits of Ω that can be produced according to the theory.

11.14. Diophantine equations and randomness

We are now ready to take the final step. A really dramatic implication of the properties of Ω is that we can exhibit a particular (exponential) Diophantine equation

$$(*) \qquad P(i, x_1, \ldots, x_m) = Q(i, x_1, \ldots, x_m)$$

such that, for each i, $(*)$ has infinitely many solutions in $x_1, \ldots, x_m$ if and only if b_i, the ith bit in Ω, equals 1. As in the definition of an exponential Diophantine relation, P and Q are functions built up from $i, x_1, \ldots, x_m$ and nonnegative integer constants by the operations of addition, multiplication and (unary) exponentiation . Denote by $(*)_i$, $i = 1, 2, \ldots$, the equation obtained from $(*)$ by fixing the parameter i. Does $(*)_i$ have infinitely many solutions? By the properties of Ω deduced above, any formal theory can answer this question for finitely many values of i only. No matter how many additional answers we learn, for instance, by experimental methods or just flipping a coin, this won't help us in any way as regards the remaining infinitely many values of i. As regards these values, mathematical reasoning is helpless, and a mathematician is not better off than a gambler flipping a coin. This holds in spite of the fact that we are dealing with basic arithmetic.

In fact, we have already developed all the technical apparatus needed to establish the above claim concerning $(*)$ and the magic bits. Recall the definition

$$\omega_n = \sum_{j=1}^{n} 2^{-|g(j)|} ,$$

where g constitutes an enumeration of programs for U that halt when started with the empty word. We fix an i and consider the ith bit $b(i, n)$ in ω_n, for increasing values of n. At first $b(i, n)$ may fluctuate irregularly between 0 and 1 but will stabilize from a certain point on. The reason for this is that because ω_n tends to Ω, $b(i, n)$ has to become b_i.

The binary relation $b(i, n) = 1$ is recursively enumerable — in fact, it is recursive. (The notation should be changed from n to x_1 to conform with $(*)$. However, no confusion should arise although we stick to the more natural n-notation.) By Theorem 11.2, there are P and Q such that $(*)$ has a solution in $x_2, \ldots, x_m$ if and only if $b(i, n) = 1$. Moreover, this *ED* representation is *singlefold*: for each i and n, there is at most one $(m-1)$-tuple $(x_2, \ldots, x_m)$ satisfying

$$(**) \qquad P(i, n, x_2, \ldots, x_m) = Q(i, n, x_2, \ldots, x_m).$$

Consequently, for each i, $b(i, n) = 1$ holds for infinitely many values of n if and only if $(**)$ holds for infinitely many m-tuples $(n, x_2, \ldots, x_m)$. Because $b_i = 1$ if and only if $b(i, n) = 1$ for infinitely many values of n, we have established the desired result, expressed in the following theorem.

Theorem 11.27. *The ith magic bit b_i equals* 1 *if and only if* $(*)$ *has infinitely many solutions.*

Chaitin, [3] has constructed $(*)$ explicitly. His equation is really huge: some $17,000$ variables and $900,000$ characters.

It is essential that we ask whether or not $(*)$ has *infinitely many* solutions for a given i. It is not sufficient to consider the *existence* of solutions because the set of (the indices of) solvable Diophantine equations is recursively enumerable and, hence, cannot lead to randomness.

Singlefoldedness could be replaced by a weaker property: for each i and n, $(**)$ holds for only finitely many $(m-1)$-tuples $x_2, \ldots, x_m$. It is not known whether the theory of Diophantine representations holds for this notion weaker than singlefoldedness. Consequently, it is not known whether P and Q in $(*)$ can be assumed to be polynomials, and thus, whether exponentiation can be avoided.

11.15. Conclusion

We end this paper with a brief philosophical conversation about the central issues discussed above. We refer the reader also to the recent article of Chaitin [4]. The final part of [1] is also very relevant in this context.

Question. A lot of things escape any given axiom system. Let us go to matters discussed in the last section above. It is hard for me to visualize that we just flip a coin to decide whether an equation has a finite or an infinite number of solutions. Both outcomes seem to be OK and in accordance with any theory we might have had before. And the additional knowledge gained by deciding about this particular equation does not help us much: infinitely many equations remain to be handled similarly.

Answer. You might have difficulties only because you forget that *everything* can be encoded as solutions of equations. All recursively enumerable sets can be represented in this way. We might equally well ask

whether a given Turing machine defines a finite or an infinite language. It might be easier for you to visualize that no formal system can answer this question in regard to all Turing machines. After all, a formal system itself is nothing but a Turing machine. There always will be machines for which the decision has to be made by coin flipping, or possibly by some experiments.

Question. I can follow all that. Still, we have a specific equation. Even if it is long, it can be written down explicitly, and it has been written down explicitly. The question whether it has a finite or an infinite number of solutions is clear enough. In my world, call it Platonic if you like, this question has a definite answer. It is only because of my ignorance that I don't know the answer. In my world the law of the excluded middle holds in great generality, at least in arithmetical matters like this. I like the idea that we always discover new things from this world, never being able to exhaust it.

Answer. Think of the infinite sequence of equations about which we have to make the decision: finitely or infinitely many solutions. So we get an infinite sequence of 0's and 1's for each completed set of decisions. Each such sequence corresponds to a (hopefully) consistent theory. There are many sequences, even if we take into account that some of them do not correspond to an axiomatizable theory (this happens if the sequence is not recursive). Which sequence is the one of your Platonic world? Is your world the same as mine? You also admitted that you don't know your world so well. Finding the answers is discovery , not creation. Objects of set theory should be discovered. You can also encode things in different ways, symbolize them as Post says. Here the physical, experimental aspect is important: which objects correspond to the universe we live in. Which of the Platonic worlds is realized, is a question of physics.

References

[1] C. Calude, *Information and Randomness. An Algorithmic Perspective. Second Edition.* Springer-Verlag, Berlin, Heidelberg, New York (2002).

[2] G. Chaitin, A theory of program size formally identical to information theory. *Journal of the Association for Computing Machinery* 22 (1975) 329–340.

[3] G. Chaitin, *Algorithmic Information Theory*. Cambridge University Press, Cambridge (1987).
[4] G. Chaitin, The Limits of Reason, *Scientific American*, March (2006) 54–61.
[5] M. Davis (ed.), *The Undecidable*. Raven Press, Hewlett, N.Y. (1965).
[6] M. Li and P. Vitanyi, Kolmogorov complexity and its applications. In: J. van Leeuven (ed.) *Handbook of Theoretical Computer Science*, Vol. A (1990) 187–254.
[7] G. Rozenberg and A. Salomaa, *Cornerstones of Undecidability*. Prentice Hall, New York, London (1994).
[8] A. Salomaa, *Formal Languages*. Academic Press, New York (1973).

Chapter 12

The Complexity of the Set of Nonrandom Numbers

Frank Stephan

Department of Mathematics and School of Computing
National University of Singapore, 2 Science Drive 2, Singapore 117543;
fstephan@comp.nus.edu.sg

Let C and H denote the plain and prefix-free description complexity, respectively. Then the sets *NRC* of nonrandom numbers with respect to C has neither a maximal nor an r-maximal superset. The set of *NRH* of nonrandom numbers with respect to H has an r-maximal but no maximal superset. Thus the lattices of recursively enumerable supersets (modulo finite sets) of *NRC* and *NRH* are not isomorphic. Further investigations deal with the related set *NRW* of numbers x with a stronger nonrandomness property: $x \neq \max\{W_e\}$ for any $e < x$ where $W_0, W_1, \ldots$ is derived from the underlying acceptable numbering of partial-recursive functions. Friedman originally asked whether $NRW \equiv_T K$ for every underlying acceptable numbering and Davie provided a positive answer for many underlying acceptable numberings. Later Teutsch asked whether the set *NRW* can be r.e. or co-r.e.; as an answer to this question it is shown that in the case that the underlying numbering is a Kolmogorov numbering, *NRW* is not n-r.e. for any n. If one uses any acceptable numbering instead of a Kolmogorov numbering, then the underlying numbering can be chosen such that *NRW* is a co-2-r.e. set; but it cannot be a 2-r.e. set for any acceptable numbering.

12.1. Introduction

Let C and H denote the plain and prefix-free description complexity, respectively. Furthermore, one can identify the numbers in the set $I_n = \{2^n-1, 2^n, 2^n+1, \ldots, 2^{n+1}-2\}$ with the binary strings of length n; a number x corresponds to a string σ iff $x+1$ is the binary value of the string 1σ. For having an easier connection to other fields of recursion theory, natural numbers are used from now on. For any number x, the n with $x \in I_n$ is called the length of x, written $|x|$. The plain description complexity C is

defined as

$$C(x) = \min\{|p| : U(p) = x\}$$

where U is a universal function. That is, the value C based on U satisfies the following two conditions:

(1) The range of U is the set of all natural numbers, so that $C(x)$ is defined for all x.
(2) For every further unary partial-recursive function V there is a constant c with $C(V(p)) \leq |p| + c$ for all p in the domain of V.

The other variant H is based on prefix-free machines. Chaitin [1–3] and Levin [11] laid the foundations and showed the significance of this alternative approach which developed together with the original notion C to the two best accepted concepts in description complexity. A prefix-free machine satisfies that all different strings p, q in its domain are incomparable with respect to the string-prefix-relation; alternatively, one can also use the Kraft-Chaitin-Theorem [1–3, 11] and say that a machine U is equivalent to a prefix-free one iff

$$\sum_{p \in dom(U)} 2^{-|p|} \leq 1.$$

A prefix-free machine U is universal iff the two conditions above hold where in Condition 2 only prefix-free machines V are considered. Then

$$H(x) = \min\{|p| : U(p) = x\}$$

where U is a prefix-free partial-recursive unary function which is universal for the class of all prefix-free partial-recursive unary functions. Now the two sets in question are defined as

- $NRC = \{x : C(x) < |x|\}$;
- $NRH = \{x : H(x) < |x|\}$.

They are called the sets of nonrandom numbers with respect to C and H or the sets of compressible strings with respect to C and H. These sets are standard examples of simple sets, which do not arise through a complicated construction but are just given as natural. Furthermore, they are wtt-complete but not btt-complete. Kummer [9] showed that NRC is tt-complete (for any given universal machine), but according to Muchnik and Peretselski [14] the tt-completeness of NRH depends on the choice of the universal machine. The existence of universal machines for which NRH

is tt-complete is easy to show; a clever definition makes it possible to satisfy the equivalence

$$n \in K \Leftrightarrow |NRH \cap \{2^n, 2^n + 1, \ldots, 2^{n+1} - 1\}| \text{ is odd.}$$

The choice of a universal machine where NRH is not tt-complete is the more difficult part.

In the present work, it is shown that the sets NRC and NRH can also be used as examples for the following theorems. Martin [13] showed that there is a recursively enumerable coinfinite set without a maximal supersets; both sets NRC and NRH have this property as well. Lachlan [10] and Robinson [18] constructed a recursively enumerable coinfinite set without an r-maximal superset; NRC is also such an example. Furthermore, Lachlan and Robinson constructed a set with an r-maximal but without any maximal superset; NRH is such an example.

For these theorems, recall that a recursively enumerable and coinfinite set A is called maximal if either $\overline{A} \subseteq^* B$ or $\overline{A} \subseteq^* \overline{B}$ for every recursively enumerable set B. Furthermore, a recursively enumerable and coinfinite set A is called r-maximal if either $\overline{A} \subseteq^* B$ or $\overline{A} \subseteq^* \overline{B}$ for every recursive set B. Here $X \subseteq^* Y$ if $X - Y$ is finite, that is, if almost all $x \in X$ are also in Y; furthermore, $X \subset^* Y$ iff $X \subseteq^* Y$ and $Y \not\subseteq^* X$.

A lot of variants of the basic notions C and H have been studied in the theory of description complexity. One related notion to the sets NRC and NRH is the set

$$NRW = \{x : \exists e < x \; [x = \max(W_e \cup \{0\})]\}$$

which is based on the underlying acceptable numbering of partial-recursive functions with W_e being the domain of the e-th partial-recursive function. $W_0, W_1, \ldots$; Friedman [6] asked what the Turing-degree of NRW is; Davie [4] obtained the by now best results with respect to this problem. Davie first solved this question for the case where the underlying numbering is a Kolmogorov numbering. Here a numbering $\varphi_0, \varphi_1, \ldots$ of partial-recursive functions is a Kolmogorov numbering iff, given any other numbering $\psi_0, \psi_1, \ldots$ of partial-recursive functions, there are constants c, d such that one can compute for every index n an index $m \leq cn + d$ with $\varphi_m = \psi_n$. Later, based on correspondence with Solovay, Davie pointed out how to solve Friedman's question for a further large class of acceptable numberings. But Friedman's question is still not completely solved. At the end, an alternative proof using Kummer's Cardinality Theorem is given for

Davie's first result that $K \leq_T NRW$ whenever the underlying numbering is a Kolmogorov numbering.

Schaefer and Teutsch also studied related problems, for example, whether NRW is an r.e. or a co-r.e. set [20]. Note that the set NRW is similar to NRC and NRH; so if one chooses the universal machines and the numbering $W_0, W_1, \ldots$ properly, then one gets $NRH \subseteq NRC \subseteq NRW$ which would directly imply that NRW is co-immune and thus not a co-r.e. set. But the obtained results are more general: There is an acceptable numbering such that — when based on this numbering — NRW is a co-2-r.e. set. But NRW cannot be a 2-r.e. set. Furthermore, if NRW is based on a Kolmogorov numbering then NRW is not n-r.e. for any number n. In any case, NRW is still ω-r.e. as witnessed by the approximation A_s with $x \in A_s$ iff there is an $y < x$ with $\{x\} \subseteq W_{y,s} \cap \{0, 1, \ldots, s\} \subseteq \{0, 1, \ldots, x\}$.

Unexplained recursion-theoretic notation follows the books of Odifreddi [15, 16] and Soare [19].

12.2. Known Results

Neither the set NRC nor the set NRH have hyperhypersimple supersets. Defining the universal machines adequately gives that $C(x) \leq H(x)$ for all x and thus $NRH \subseteq NRC$; this will be assumed here and in all further proofs.

Remark 12.1. Martin [13] proved that there is a recursively enumerable coinfinite set without a maximal superset. Odifreddi [16] proved Proposition IX.2.27 by producing a set A which contains for all n and all $e \leq n$ every element of a set $I_n - W_e$ whenever that set has at most one element y. He then showed that such a set A has no maximal superset. As

$$C(y) \leq C((n+e)(n+e+1)+n) + c \leq 2\log(n+1)$$

for some constant c independent of n, e whenever such a y exists, the construction implies that for every function f with $f(n) \geq 3 \cdot \log(n)$ for almost all n, the sets

$$\{x : C(x) < f(x)\} \text{ and } \{x : H(x) < f(x)\}$$

do not have a maximal superset. In particular NRC and NRH do not have maximal supersets. One can generalize the above result to all recursive, increasing and unbounded functions f by using larger intervals J_n instead of I_n. Taking $J_n = \{x : f(x) \in I_n\}$, one gets that the complement of every maximal set contains infinitely many x with $C(x) < f(x)$.

So this result is already more or less there. Nevertheless, a formal proof will be given for the sake of completeness. The description complexity part of this proof can be put into the following form, which is quite often used implicitly although the author is not aware of a direct reference.

Proposition 12.1. *Let A be a recursively enumerable set. Then there is a constant c such that the following holds:*

If $NRC \subseteq A$ then for all n, either $I_n \subseteq A$ or $|I_n - A| \geq 2^{-c} \cdot |I_n|$.

If $NRH \subseteq A$ then for all n, either $I_n \subseteq A$ or $|I_n - A| \geq 2^{-2 \cdot |n| - c} \cdot |I_n|$.

Proof. Assume that A is recursively enumerable and $NRC \subseteq A$.

Then one splits each interval I_n into intervals $J_{n,m}$ with 2^{n-m} elements for $m = 1, 2, \ldots, n$ plus one further element. Now one can define a partial-recursive function V such that $V[J_{n,m}] = \{V(p) : p \in J_{n,m}\} \supseteq I_{n+m} - A$ whenever $|I_{n+m} - A| \leq 2^{n-m}$ by waiting on each interval $J_{n,m}$ for the event $|I_{n+m} - A| \leq |J_{n,m}|$ to take place and by then assigning the values correspondingly. Note that V is not total as there are intervals $J_{n,m}$ where the corresponding event will never take place. It follows that $C(x) \leq n - m$ for all $x \in I_{n+m}$ whenever $|I_{n+m} - A| \leq 2^{n-m}$. Now one can argue that there is a constant c' with $C(V(p)) < |p| + c'$ for all p in the domain of V. It follows that $I_{n+c'}$ is not in the range of $V[J_{n,c}]$ and hence there is no n such that $1 \leq |I_{n+c'} - A| \leq 2^{n-c'}$.

The second result is just based on the fact [12] that $|H(x) - C(x)| \leq 2 \cdot |n| + c''$ for some constant c'', all n and all $x \in I_n$. So the whole theorem holds with $c = 2(c' + c'')$. □

Proposition 12.2. *Neither the set NRC nor the set NRH have a hyper-hypersimple superset.*

Proof. As $NRH \subseteq NRC$, it is sufficient to show that no coinfinite r.e. superset A of NRH is hyperhypersimple. So let such a coinfinite r.e. superset A of NRH be given. By Proposition 12.1 there is an increasing and unbounded recursive function b such that for infinitely many n the set $I_n - A$ contains at least $b(n)$ many elements.

Having this property, it is easy to define a partial-recursive function ψ such that ψ takes on every set I_n with $|I_n - A| \geq b(n)$ on $b(n)$ elements of $I_n - A$ the values $0, 1, \ldots, b(n) - 1$, respectively. Whenever then some $x \in I_n$ with $\psi(x) = m \wedge m < b(n)$ is enumerated into A_{s+1} at stage s, one takes the least value $x \in I_n - A_{s+1}$ where $\psi_s(x)$ is still undefined and

assigns the value $\psi_{s+1}(x) = m$.

By choice of the function b there are for every m infinitely many n such that $|I_n - A| \geq n > m$ and $\psi(x) = m$ for an $x \in I_n - A$. It follows from a result of Yates [21] that A is not hyperhypersimple. □

The next result is an alternative proof for the result of Robinson [18] in 1967 and Lachlan [10] in 1968 that there is a recursively enumerable and coinfinite set without an r-maximal superset. The proof is based on an adjustment of Lachlan's proof [10].

Theorem 12.1. *NRC has no r-maximal superset.*

Proof. Let A be a recursively enumerable and coinfinite superset of *NRC*. By Proposition 12.1 there is a constant c such that, for infinitely many n, $I_n - A$ has at least 2^{n-c} elements. Now let

$$B_m = \{m, m + 2^{c+1}, m + 2 \cdot 2^{c+1}, m + 3 \cdot 2^{c+1}, \ldots\}$$

for $m = 0, 1, \ldots, 2^{c+1} - 1$. These sets form a finite partition of the natural numbers, but for each m, $\overline{A} \not\subseteq^* B_m$ as

$$\exists^\infty n \, [\, |I_n - A| > 2^{-c-1} \cdot |I_n| \geq |B_m \cap I_n| \,].$$

Hence A is not r-maximal. □

12.3. *NRH* has r-maximal supersets

One main result of the present work is that in contrast to *NRC*, *NRH* has an r-maximal superset.

Theorem 12.2. *There is an r-maximal superset of NRH.*

Proof. An r-maximal set A is constructed such that $A \cup NRH$ is coinfinite. Then $A \cup NRH$ is also r-maximal and a superset of *NRH*.

Now let $J_0, J_1, J_2, \ldots$ be a partition of the natural numbers such that each J_n has 2^{3n} elements. Let $L_n = \cup\{I_m : m \in J_n\}$ be the union of all I_m such that $m \in J_n$. The sets $L_0, L_1, L_2, \ldots$ are also a recursive partition of the natural numbers.

First, a recursively enumerable set B is defined such that for all r.e. sets W_i, W_j and $n > i + j$, if $L_n - B \subseteq W_i \cup W_j$ then either $L_n - B \subseteq W_i$ or $L_n - B \subseteq W_j$. This is obtained by making one indirect step into Lachlan's construction. More precisely, $L_n - B$ is forced to become a subset of W_e whenever W_e contains at stage s the majority of the sets I_k with $I_k \not\subseteq B_s$

satisfy that at least the half of $I_k - B_s$ is contained in $W_{e,s}$. More formally, the following is done.

- For each n and stage s, let e be the remainder of s divided by n and do the following:
- Let $D_0 = \{k \in J_n : I_k \not\subseteq B_s\}$;
- Let $D_1 = \{k \in D_0 : |(I_k - B_s) \cap W_{e,s}| \geq 0.5 \cdot |I_k - B_s|\}$;
- If $2|D_1| \geq |D_0|$ and $L_n - B_s \not\subseteq W_{e,s}$ then update B_s on L_n by letting
  - $B_{s+1} \cap I_k = I_k$ for all $k \in D_0 - D_1$;
  - $B_{s+1} \cap I_k = (B_s \cap I_k) \cup (I_k - W_{e,s})$ for all $k \in D_1$;

  else let $B_{s+1} \cap L_n = B_s \cap L_n$.

Now one can verify that the desired property holds. Assume that $L_n \not\subseteq B$ but $L_n \subseteq W_i \cup W_j$ and t is so large that $B_t \cap L_n = B \cap L_n$, $W_{i,t} \cap L_n = W_i \cap L_n$ and $W_{j,t} \cap L_n = W_j \cap L_n$. In every stage $s \geq t$, $D_0 = \{k \in J_n : I_k \not\subseteq B\}$ as $B_s \cap L_n = B \cap L_n$. Furthermore,

$$\forall k \in D_0 \; \exists e \in \{i, j\} \; [\, |(I_k - B_s) \cap W_{e,s}| \geq 0.5 \cdot |I_k - B_s| \,].$$

Thus, either for $e = i$ or for $e = j$, it must hold that at stage $s = tn + e$ that $2|D_1| \geq |D_0|$. As at this step, no new elements are put into B, the condition $L_n - B_s \subseteq W_{e,s}$ must be satisfied. So $L_n - B \subseteq W_i$ or $L_n - B \subseteq W_j$.

Second, it is shown that for almost every n, $L_n \not\subseteq B$. To see this, note that for each interval L_n it happens only in at most n stages s that some elements of L_n are enumerated into B. One can easily verify that after each step, at least half of the $k \in J_n$ with $I_k \not\subseteq B_s$ also satisfy the condition $|I_k B_{s+1}| \geq |I_k - B_s| \cdot 0.5$ while all other $k \in J_n$ satisfy $I_k \subseteq B_{s+1}$. As a consequence, in the limit, $|J_n| \cdot 2^{-n}$ of the intervals I_k with $k \in J_n$ satisfy $|I_k - B| \geq |I_k| \cdot 2^{-n} \geq 2^{k-n}$. As $|J_n| = 2^{3n}$, the number of these intervals is at least 2^{2n}.

Chaitin [2] showed that there is a constant c independent of k and r with $H(x) \leq k + H(k) - r$ for at most 2^{k+c-r} numbers $x \in I_k$; Downey and Hirschfeldt [5] call it the "Counting Theorem". Taking $r = n + c$ gives that at least $2^k - 2^{k-n}$ members x of I_k satisfy $H(x) > k + H(k) - n - c$ and thus $H(x) > k + n - c$. So, for sufficiently large n, the set J_n contains a k such that $H(k) \geq 2n$ and $I_k \not\subseteq B \cup NRH$. Hence $L_n \not\subseteq B \cup NRH$ for all sufficiently large n.

Third, taking any maximal set M and $A = B \cup \{x : \exists n \in M \, (x \in L_n)\}$, the sets A and $A \cup NRH$ are r-maximal. Clearly, A and $A \cup NRH$ are

coinfinite by the preceding two paragraphs. For each e define the r.e. set

$$V_e = \{n : L_n - B \subseteq W_e\}.$$

Now let i, j be indices of r.e. sets such that W_j is the complement of W_i. By the first part of the proof, all numbers $n > i + j$ are in $V_i \cup V_j$. As M is maximal, $\overline{M} \subseteq^* V_e$ for some $e \in \{i, j\}$. Then $\overline{A} \subseteq^* W_e$. Therefore, A and $A \cup NRH$ are r-maximal. □

The r-maximal set $A \cup NRH$ constructed has by Theorem 12.2 no hyper-hypersimple superset. Thus it has no maximal superset. Such type of r-maximal sets had been constructed by Robinson [18] in 1967 and by Lachlan [10] in 1968.

Corollary 12.1. *There is an r-maximal set without a maximal superset.*

A maximal set A has the property that the structure $\mathcal{L}^*_A$ of its r.e. superset modulo finite sets is just the 2-element Boolean Algebra. Hyperhypersimple sets are characterised as those sets where this structure is a Boolean Algebra. Theorems 12.1 and 12.2 show that the two sets of nonrandom numbers have a different superset structure.

Corollary 12.2. *Given a set E, let $(\mathcal{L}^*_E, \subset^*)$ be the partially ordered set of the r.e. supersets of E modulo finite sets with the ordering induced from set-inclusion. Then the structures $(\mathcal{L}^*_{NRC}, \subset^*)$ and $(\mathcal{L}^*_{NRH}, \subset^*)$ are not isomorphic.*

12.4. The Problems of Friedman and Teutsch

A numbering φ of partial recursive functions is called a Kolmogorov numbering iff for every further numbering of partial-recursive functions ψ there are constants c, d such that, for all x, every ψ_x equals some φ_y with $y < cx + d$. This translated of course to the domains W_x of φ_x: for every further numbering $A_0, A_1, A_2, \ldots$ of any r.e. sets there are constants c, d such that for every x there is an $y \leq cx + d$ with $W_y = A_x$. Furthermore, if $W_0, W_1, W_2, \ldots$ is derived from an acceptable numbering φ of partial recursive functions, then there is for every numbering $A_0, A_1, A_2, \ldots$ of any r.e. sets a recursive function f such that $W_{f(x)} = A_x$ for all indices x. Let NRW denote the set

$$NRW = \{x : \exists y < x \ [x = \max(W_y \cup \{0\})]\}$$

for a given fixed Kolmogorov numbering φ of the partial recursive functions. Friedman [6] asked whether $NRW \equiv_T K$ for every underlying acceptable numbering φ; certainly it is easy to construct some for which is true. Teutsch [20] added the question whether *NRW* can be r.e. or co-r.e. for some acceptable numbering. The general case is indeed the more difficult one and therefore only Teutsch's question can be answered for all acceptable numberings.

In the following, a set is 2-r.e. iff it is the difference of two r.e. sets, it is 3-r.e. iff it is the difference of an r.e. set minus a 2-r.e. set and so on. Alternatively, one can say that a set A is n-r.e. iff there is an approximation A_s of A such that $A_0 = \emptyset$ and there are for every n at most n indices with s with $A_{s+1}(x) \neq A_s(x)$. Recall the default approximation $A_0, A_1, A_2, \ldots$ is that

$$x \in A_s \Leftrightarrow \exists y < x \; [\{x\} \subseteq W_{y,s} \cap \{0,1,\ldots,s\} \subseteq \{0,1,\ldots,x\}].$$

This approximation makes for each x up to $2x$ mind changes; so it witnesses that *NRW* is ω-r.e. but it does not witness that *NRW* is n-r.e. for any natural number n. So the main question is whether *NRW* can be n-r.e. for some n. The answer is affirmative if the underlying numbering is not requested to be a Kolmogorov numbering.

Theorem 12.3. *There is an acceptable numbering such that NRW is the complement of some 2-r.e. set.*

Proof. Fix the recursive partition $J_0, \{x_0\}, J_1, \{x_1\}, J_2, \{x_2\}$ of the natural numbers satisfying $|J_k| = 2^k + 2$ for all k; this is the partition given by $J_0 = \{0,1,2\}$, $x_0 = 3$, $J_1 = \{4,5,6,7\}$, $x_1 = 8$, $J_2 = \{9,10,11,12,13,14\}$ and so on. Furthermore, let for every k the number

$$y_k = \min\{x \in J_k : \forall z \in J_k \; [z \leq x \vee H(z) < k]\}.$$

Note that the intervals J_k are chosen so large that every interval J_k contains at least three numbers z with $H(z) \geq k$ and y_k is just the maximum of all these z. Furthermore, the y_k are uniformly approximable from above as the formula of their definition shows, let $y_{k,s}$ be the value of y_k before stage s. Now define $A_s(x) = 1$ iff there is a k with $x \in J_k$ and $s > 0$ and $x = y_{k,s}$. It is easy to verify that this approximation witnesses that A is 2-r.e.: At the beginning, $A_s(x) = 0$; if $y_{k,s}$ has come down and reached x then $A_s(x) = 1$; if $y_{k,s}$ has gone below x then $A_s(x) = 0$ again.

The proof is completed by defining an acceptable enumeration $W_0, W_1, \ldots$ such that *NRW* based on this numbering is a finite variant

of $\overline{A}$. So let any acceptable numbering ψ be given. Define W_{x_k} to be the domain of ψ_k in order to make the numbering $W_0, W_1, \ldots$ acceptable. For $x \in J_k$, let

$$W_x = \begin{cases} \{x+1\}, & \text{if } y_k > x+1, \\ \{x+1, x+2\}, & \text{otherwise.} \end{cases}$$

It is straightforward to verify that $W_0, W_1, \ldots$ is indeed a numbering, the only important ingredient is that the y_k are approximated from above.

Let $x > 0$ be given and choose the k with $\min(J_k)+1 \leq x \leq \max(J_k)+2$. If $x < y_k$ then $x < \max(J_k)$, $x-1 \in J_k$, $y_k > (x-1)+1$ and $W_{x-1} = \{x\}$, hence $x \in NRW$. If $x > y_k$ then $x \geq \min(J_k)+2$ (as there are three or more numbers $z \in J_k$ with $H(z) \geq k$), $x-2 \in J_k$, $y_k \leq (x-2)+1$ and $W_{x-2} = \{x-1, x\}$, hence $x \in NRW$ again. If $x = y_k$ then $x \in NRW$ only if there is an $\ell \leq k$ with W_{x_ℓ} being nonempty and having the maximum y_k. If this holds, one can compute the value of y_k from k, ℓ by waiting for the first stage s where $y_{k,s} \in W_{x_\ell,s}$; then the value $y_{k,s}$ equals y_k. It follows that $H(y_k) \leq H(k) + H(\ell) + c$ for some constant c independent of k and ℓ. As $H(k)$ and $H(\ell)$ are both logarithmic in k and $H(y_k) \geq k$ for all k, this can happen only for finitely many k. In other words, $y_k \notin NRW$ for almost all k.

This shows that the complement of NRW is a finite variant of A. By adjusting the co-2-r.e. approximation to $\overline{A}$ at finitely many places, one receives a co-2-r.e. approximation to NRW. □

The next result shows that NRW can never by 2-r.e.; as all r.e. and co-r.e. sets are 2-r.e., this result provides a negative answer to Teutsch's question [20].

Theorem 12.4. *For every underlying acceptable numbering φ, the set NRW is not 2-r.e., that is, NRW is not the difference of two r.e. sets.*

Proof. Assume by way of contradiction that $NRW = W_i - W_j$. Note that NRW is coinfinite as there are infinitely many indices of the empty set. Using the fixed-point theorem, one can construct sets W_a, W_b such that

$$W_a = \{x : \forall y < x \ [y \in W_i \cup \{0, 1, \ldots, a\}]\};$$
$$W_b = \{x : \exists s \ [x > b \wedge x \in W_{j,s+1} \wedge W_{j,s} \subseteq \{0, 1, \ldots, b\}]\}.$$

If W_j is infinite, then the maximum of W_b is greater than b and is in W_j. But then this maximum is a member of NRW but not of $W_i - W_j$. Hence

W_j must be finite and W_i coinfinite. So the set W_a is finite as its maximum is the least nonelement of W_i which is greater than a. Again this maximum is a member of NRW but not of $W_i - W_j$. This gives a contradiction to $NRW = W_i - W_j$. Hence NRW is not a 2-r.e. set. □

In the case that the underlying numbering is a Kolmogorov numbering, a even stronger result can be obtained: NRW is not n-r.e. for any natural number n. The proof uses a basic fact stated in the following remark.

Remark 12.2. If the underlying numbering is a Kolmogorov numbering then there is a constant c such that $|\{0, 1, \ldots, cx\} - NRW| > x$ for all x.

To see this, note that $0 \notin NRW$, hence it is enough to look at positive x. Let B_m be the complement of $\{m\}$ for all m. Then there are constants c', d' such that for each B_m equals a set W_n with $n < c'm + d'$. So for every $x > 0$ there are $x+1$ indices of sets B_m with $m \leq x$ below $(c'+d')x$. Therefore at most $(c'+d')x - x - 1$ of the numbers $0, 1, \ldots, (c'+d')x$ are indices of finite sets. Hence, only $(c'+d')x - x - 1$ of the numbers $0, 1, \ldots, (c'+d')x$ can be in NRW. In other words, taking $c = c' + d'$ implies the desired inequality $|\{0, 1, \ldots, cx\} - NRW| > x$ for all x.

Theorem 12.5. *If the underlying numbering is a Kolmogorov numbering then the set NRW is not n-r.e. for any n.*

Proof. Let c be the constant from Remark 12.2. Assume by way of contradiction that there is a number n and an approximation $A_0, A_1, A_2, \ldots$ to NRW which makes at most n mind changes for all x. Now one construct a family $B_0, B_1, \ldots$ of sets such that, using the fixed point theorem, one knows the constants c', d' to translate the B_x into some W_y with $y \leq c'x + d'$. Now let $B_{x,0} = \{c'x + d' + 1\}$. At stage s let $y = \max(B_{x,s})$ and check whether the following conditions hold:

- $y \in A_s$;
- $|\{0, 1, \ldots, 3yc\} - A_s| \geq 3y$;
- $\max(B_{u,s}) > 3cy$ for all $u < x$.

If these conditions are satisfied then let $B_{x,s+1} = B_{x,s} \cup \{y + 1\}$ else let $B_{x,s+1} = B_{x,s}$.

If some of the so constructed sets is finite, then choose x to be the least number such that B_x is finite and let $y = \max(B_x)$. As there is a $z \leq c'x + d'$ with $W_z = B_x$ and $y \geq c'x + d' + 1 > z$, it holds that $y = \max(W_z)$ and $y \in NRW$. There is a stage s which is so large that

- $y = \max(B_{x,s})$;
- $A_s(z) = NRW(z)$ for all $z \in \{0, 1, \ldots, 3cy\}$; in particular, $y \in A_s$ and $|\{0, 1, \ldots, 3yc\} - A_s| \geq 3y$.
- for every $u < x$, there is an element larger than $3cy$ of the infinite set B_u enumerated, that is, $\max(B_{u,s}) > 3cy$.

Now it would follow that $B_{x,s+1} = B_{x,s} \cup \{y+1\}$ by the construction of the set B_x in contradiction to the assumption that $y = \max(B_x)$. Hence B_x has to be infinite.

Given n, let m be so large that every $x \leq 2cn + n + 1$ satisfies the condition $\max(B_{x,0}) < m$. For these x, let s_x be the first stage where $m+1 \in B_{x,s_x+1}$. As $\max(B_{x,s_{x+1}}) > 3mc$ and $|\{0, 1, \ldots, 3mc\} - A_{s_x}| \geq 3m$, there is for any $y \in \{m, m+1, \ldots, 3mc\}$ a stage $t \in \{s_x, s_x+1, \ldots, s_{x+1}-1\}$ with $y = \max(B_{x,t})$ and $y < \max(B_{x,t+1})$, hence $y \in A_t$ for that t. Hence there at least $2m$ numbers $y \in \{m, m+1, \ldots, 3cm\}$ for which there is a $t \in \{s_x, s_x + 1, \ldots, x_{x+1} - 1\}$ with $A_t(y) = 0 < A_{t+1}(y) = 1$. As this applies to $x = 0, 1, \ldots, 2cn + n$, there are $2m(2cn + n)$ pairs (y, t) with $y \in \{m, m+1, \ldots, 3cm\}$ and $A_t(y) < A_{t+1}(y)$. So there is an number y with $A_t(y) < A_{t+1}(y)$ for at least $2n$ stages t as $|\{m, m+1, \ldots, 3cm\}| = 2cm+1$ and

$$\frac{2m(2cn+n)}{2mc+1} \geq \frac{2mn(2c+1)}{(2c+1)m} = 2n.$$

This contradicts to NRW being an n-r.e. set. □

The last result of this paper gives a short alternative proof for Davie's first result that whenever NRW is based on a Kolmogorov numbering then $NRW \equiv_T K$. As NRW is ω-r.e., obviously $NRW \leq_T K$ and this direction is not included in the proof. After his first result, Davie corresponded with Solovay who indicated to him how to generalize his first result such that it answers Friedman's question for every "usual" underlying acceptable numbering where "usual" means that there exists a polynomial p such that for every recursively enumerable family $A_0, A_1, \ldots$ of r.e. sets and every x there is an $y < p(x)$ with $W_y = A_x$. The proof for Davie's first result given here uses Kummer's Cardinality Theorem [7, 8], but it should be noted that also Owing's preliminary results [17] are sufficient to prove this result as the sets E_n in the proof below are uniformly recursive relative to NRW.

Theorem 12.6. *If the underlying numbering is a Kolmogorov numbering then the set NRW is Turing complete; that is, $NRW \equiv_T K$.*

Proof. Note that the constant c from Remark 12.2 satisfies for every x the condition $\{x, x+1, \ldots, cx\} \not\subseteq NRW$. Now let $J_0, J_1, \ldots$ be a partitioning of the natural numbers into intervals satisfying $\min(J_n)c < \max(J_n)$ for all natural numbers n. Furthermore, let $D_0, D_1, D_2, \ldots$ be a canonical indexing of all finite sets of natural numbers with $D_0 = \emptyset$. Define

$$B_m = \{y : y \leq m \cdot (|D_n| + 1) + |K \cap D_n| \text{ for the } n \text{ with } m \in J_n\}$$

and note that the B_m are uniformly r.e. finite sets. Now choose a constant k such that for all n with $|D_n| \geq k$ and all $m \in J_n$, every set B_m has an index u such that $W_u = B_m$ and $u < mk \leq m \cdot (|D_n| + 1) + |K \cap D_n|$. The constant exists since $W_0, W_1, W_2, \ldots$ is a Kolmogorov numbering of r.e. sets. Now one can define the following sets in a way that they are uniformly r.e. relative to NRW:

$$E_n = \{u \in \{0, 1, \ldots, k\} : \forall m \in J_n \ [\, |D_n| = k \wedge mk + u \in \overline{NRW}\,]\}.$$

By choice, E_n contains $|K \cap D_n|$ whenever $|D_n| = k$. Furthermore, $c(k+1)\min(J_n) < (k+1)\max(J_n)$, thus there is for every n some $u \in \{0, 1, \ldots, k\}$ not contained in E_n. Now applying Kummer's Cardinality Theorem [7, 8] to the cardinality function $\#_k^K$ gives $K \leq_T NRW$. □

Acknowledgments: The author wants to thank Martin Kummer for useful discussions and detailed comments. The author also thanks George Davie, Carl Jockusch, Marcus Schaefer and Jason Teutsch for correspondence; Jason Teutsch permitted him to see an incomplete version of his thesis including the open problems there.

References

[1] Gregory J. Chaitin. *A theory of program size formally identical to information theory. Journal of the Association for Computing Machinery*, 22:329–340, 1975.

[2] Gregory J. Chaitin. Information-theoretic characterizations of recursive infinite strings. *Theoretical Computer Science*, 2:45–48, 1976.

[3] Gregory J. Chaitin. Algorithmic information theory, *IBM Journal of Research and Development*, 21:350–359+496, 1977.

[4] George Davie. Foundations of Mathematics – Recursion Theory Question. `http://cs.nyu.edu/pipermail/fom/2002-May/005535.html`, This posting answers Friedman's Question [6].

[5] Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic Randomness and Complexity.* Manuscript, 2007.

[6] Harvey Friedman. Foundations of Mathematics – Recursion Theory Question. http://cs.nyu.edu/pipermail/fom/2002-February/005289.html, 2002.

[7] Valentina Harizanov, Martin Kummer and James C. Owings, Jr. Frequency computation and the cardinality theorem. *Journal of Symbolic Logic*, 57:682–687, 1992.

[8] Martin Kummer. A proof of Beigel's cardinality conjecture. *The Journal of Symbolic Logic*, 57:677–681, 1992.

[9] Martin Kummer. On the complexity of random strings (extended abstract). STACS 1996, *Springer LNCS*, 1046:25-36, 1996.

[10] Alistair H. Lachlan. On the lattice of recursively enumerable sets. *Transactions of the American Mathematical Society*, 130:1–37, 1968.

[11] Leonid A. Levin. *Some Theorems on the Algorithmic Approach to Probability Theory and Information Theory*. Dissertation in Mathematics, Moscow, 1971.

[12] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Section 3.5, Second Edition, Springer, 1997.

[13] Donald A. Martin. A theorem on hyperhypersimple sets. *The Journal of Symbolic Logic*, 28:273–278, 1963.

[14] Andrei A. Muchnik and Semen Ye. Positselsky. Kolmogorov entropy in the context of computability theory. *Theoretical Computer Science*, 271:15–35, 2002.

[15] Piergiorgio Odifreddi. *Classical Recursion Theory*, *Studies in Logic and the Foundations of Mathematics*, volume 125. North-Holland, Amsterdam, 1989.

[16] Piergiorgio Odifreddi. *Classical Recursion Theory II*, *Studies in Logic and the Foundations of Mathematics*, volume 143. Elsevier, Amsterdam, 1999.

[17] James C. Owings, Jr. A cardinality version of Beigel's Nonspeedup Theorem. *Journal of Symbolic Logic*, 54:761–767, 1989.

[18] Robert W. Robinson. Simplicity of recursively enumerable sets. *The Journal of Symbolic Logic*, 32:167–172, 1967.

[19] Robert I. Soare. *Recursively enumerable sets and degrees*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.

[20] Jason Teutsch. *Noncomputable Spectral Sets*, Question A.5, Draft of forthcoming PhD Thesis, private communication, 2007.

[21] C. E. Mike Yates. Recursively enumerable sets and retracing functions. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 8:331–345, 1962.

Chapter 13

Omega and the Time Evolution of the n-Body Problem

Karl Svozil

Institut für Theoretische Physik, University of Technology Vienna, Wiedner Hauptstraße 8-10/136, A-1040 Vienna, Austria;

svozil@tuwien.ac.at

The series solution of the behavior of a finite number of physical bodies and Chaitin's Omega number share quasi-algorithmic expressions; yet both lack a computable radius of convergence.

13.1. Solutions to the n–body problem

The behaviour and evolution of a finite number of bodies is a sort of "rosetta stone" of classical celestial mechanics insofar as its investigation induced a lot of twists, revelations and unexpected issues. Arguably the most radical deterministic position on the subject was formulated by Laplace, stating that [1, Chapter II] *"We ought then to regard the present state of the universe as the effect of its anterior state and as the cause of the one which is to follow. Given for one instant an intelligence which could comprehend all the forces by which nature is animated and the respective situation of the beings who compose it an intelligence sufficiently vast to submit these data to analysis it would embrace in the same formula the movements of the greatest bodies of the universe and those of the lightest atom; for it, nothing would be uncertain and the future, as the past, would be present to its eyes."*

In what may be considered as the beginning of deterministic chaos theory, Poincaré was forced to accept a gradual departure from the deterministic position: sometimes small variations in the initial state of the bodies could lead to huge variations in their evolution in later times. In Poincaré's own words [2, Chapter 4, Sect. II, pp. 56-57], *"If we would know the laws of Nature and the state of the Universe precisely for a certain time, we would*

be able to predict with certainty the state of the Universe for any later time. But [[. . .]] it can be the case that small differences in the initial values produce great differences in the later phenomena; a small error in the former may result in a large error in the latter. The prediction becomes impossible and we have a 'random phenomenon.' "

In what follows we present an even more radical departure from Laplacian determinism. A physical system of a finite number of bodies capable of universal computation will be presented which has the property that certain propositions remain not only provable intractable, but provable unknowable. Pointedly stated, our knowledge of any such system remains incomplete forever. For the sake of making things worse, we shall "compress" and "compactify" this kind of physical incompleteness by considering physical observables which are truly random, i.e., algorithmically incompressible and stochastic.

The methods of construction of physical n–body observables exhibiting the above features turn out to be rather humble and straightforward. In a first step, it suffices to reduce the problem to the halting problem for universal computation. This can be achieved by "embedding" a universal computer into a suitable physical system of a finite number of bodies. The associated ballistic computation will be presented in the next section. In a second reduction step, the universal computer will be directed to attempt to "compute" Chaitin's Omega number, which is provable random, and which is among the "most difficult" tasks imaginable. Finally, consequences for the series solutions [3–6] to the general n-body problem will be discussed.

13.2. Reduction by ballistic computation

In order to embed reversible universal computation into a quasi-physical environment, Fredkin and Toffoli introduced a "billiard ball model" [7–10] based on the collisions of spheres as well as on mirrors reflecting the spheres. Thus collisions and reflections are the basic ingredients for building universal computation.

If we restrict ourselves to classical gravitational potentials without collisions, we do not have any repulsive interaction at our disposal; only attractive $1/r$ potentials. Thus the kinematics corresponding to reflections and collisions has to be realized by purely attractive interactions. Fig. 13.1a) depicts a Fredkin gate realized by attractive interaction which corresponds to the analogue billiard ball configuration achieved by collisions (e.g., [8, Fig. 4.5]). At points **A** and **B** and time t_i, two bodies are either put at both

locations **A** and **B**; or alternatively, one body is put at only one location, or no bodies are placed at all. If bodies are present at both **A** and **B**, then they will follow the right paths at later times t_f. In case only one body is present at **A** or **B**, only one of the dotted inner outgoing paths will be used. Boolean logic can be implemented by the presence or absence of balls. Fig. 13.1b) depicts a reflective "mirror" element realized by a quasi-steady mass. For a proof of universality, we refer to the classical papers on the

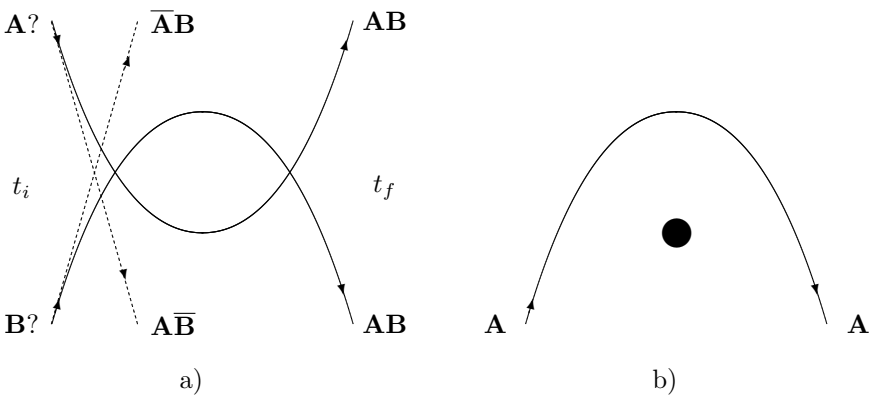


Figure 13.1. Elements of universal ballistic computation realized by attractive $1/r$ potentials. a) Fredkin's gate can perform logical reversibility: bodies will appear on the right outgoing paths if and only if bodies came in at both **A** and **B**; b) Reflective "mirror" element realized by a quasi-steady mass.

billiard ball model cited above.

13.3. Undecidability and Omega in the n-body problem

By reduction to the recursive unsolvability of the rule inference [11–15] and the halting [16–18] problems, the general induction and forecasting problem of the n-body ballistic universal computer sketched above is provable unsolvable. That is, there exist initial configurations for which it is impossible to predict with certainty whether or not certain "final" states will eventually be reached. Moreover, given a finite segment of the time evolution alone is in general insufficient for a derivation of the initial state configuration of the n-body problem.

For the sake of making things worse, we imagine an n-body system

attempting to evaluate its associated halting probability Omega [19–21]. In order to establish the equivalent of prefix-free programs, only a limited number of n-body initial configurations contribute to the configuration. Furthermore, as the computation is reversible and procedural, certain "final" configurations must be defined as halting states. This is a feature shared with the billiard ball model, as well as with quantum computation.

13.4. Consequences for series solutions

Wang's power series solution to the n-body problem [4, 6] may converge "very slowly" [5]. Indeed, by considering the halting problems above, and in particular by reduction to the computation of the halting probability Omega, certain physical observables associated with the n-body problem do not have a power series solution with a *computable radius of convergence*.

This is a particular case of Specker's theorems in recursive analysis, stating that there exist recursive monotone bounded sequences of rational numbers whose limit is no computable number [22]; and there exist a recursive real function which has its maximum in the unit interval at no recursive real number [23].

It is important to realize that, while it may be possible to evaluate the state of the n bodies by Wang's power series solution for any finite time with a computable, though excessively large, radius of convergence, global observables, referring to all times, may be uncomputable. Examples of global observables are, for instance, associated with the stability of the solar system and associated with it, bounds for the orbits.

This, of course, stems from the metaphor and robustness of universal computation and the capacity of the n-body problem to implement universality. It is no particularity and peculiarity of Wang's power series solution. Indeed, the troubles reside in the capabilities to implement Peano arithmetic and universal computation by n-body problems. Because of this capacity, there cannot exist other formalizable methods, analytic solutions or approximations capable to decide and compute certain decision problems or observables for the n-body problem.

Chaitin's Omega number, the halting probability for universal computers, has been invented in a totally different, unrelated algorithmic context, and with intentions in mind which are seemingly different from issues in classical mechanics. Thus it is fascinating that Omega is also relevant for the prediction of the behaviour and the movement of celestial bodies.

References

[1] P.-S. Laplace, *Philosophical Essay on Probabilities*. Translated from the fifth French edition of 1825. (Springer, Berlin, New York, 1995,1998). ISBN 978-0-387-94349-7. URL `http://www.archive.org/details/philosophicaless00lapliala`.

[2] H. Poincaré, *Wissenschaft und Hypothese*. (Teubner, Leipzig, 1914).

[3] K. E. Sundman, Memoire sur le problème de trois corps, *Acta Mathematica*. **36**, 105–179, (1912).

[4] Q. D. Wang, The global solution of the n-body problem, *Celestial Mechanics*. **50**, 73–88, (1991). doi: 10.1007/BF00048987. URL `http://dx.doi.org/10.1007/BF00048987`.

[5] F. Diacu, The solution of the n-body problem, *The Mathematical Intelligencer*. **18**(3), 66–70, (1996).

[6] Q. D. Wang, Power series solutions and integral manifold of the n-body problem, *Regular & Chaotic Dynamics*. **6**(4), 433–442, (2001). doi: 10.1070/RD2001v006n04ABEH000187. URL `http://dx.doi.org/10.1070/RD2001v006n04ABEH000187`.

[7] E. Fredkin and T. Toffoli, Conservative logic, *International Journal of Theoretical Physics*. **21**(3-4), 219–253, (1982). doi: 10.1007/BF01857727. URL `http://dx.doi.org/10.1007/BF01857727`. reprinted in [24, Part I, Chapter 3].

[8] N. Margolus, Physics-like model of computation, *Physica*. **D10**, 81–95, (1984). reprinted in [24, Part I, Chapter 4].

[9] N. Margolus. Universal cellular automata based on the collisions of soft spheres. In ed. A. Adamatzky, *Collision-based computing*, pp. 107–134. Springer, London, (2002). URL `http://people.csail.mit.edu/nhm/cca.pdf`.

[10] A. Adamatzky. New media for collision-based computing. In ed. A. Adamatzky, *Collision-based computing*, pp. 411–442. Springer, London, (2002). URL `http://people.csail.mit.edu/nhm/cca.pdf`.

[11] E. M. Gold, Language identification in the limit, *Information and Control*. **10**, 447–474, (1967). doi: 10.1016/S0019-9958(67)91165-5. URL `http://dx.doi.org/10.1016/S0019-9958(67)91165-5`.

[12] L. Blum and M. Blum, Toward a mathematical theory of inductive inference, *Information and Control*. **28**(2), 125–155 (June, 1975).

[13] D. Angluin and C. H. Smith, A survey of inductive inference: Theory and methods, *Computing Surveys*. **15**, 237–269, (1983).

[14] L. M. Adleman and M. Blum, Inductive inference and unsolvability, *The Journal of Symbolic Logic*. **56**, 891–900 (Sept., 1991). doi: 10.2307/2275058. URL `http://dx.doi.org/10.2307/2275058`.

[15] M. Li and P. M. B. Vitányi, Inductive reasoning and Kolmogorov complexity, *Journal of Computer and System Science*. **44**, 343–384, (1992). doi: 10.1016/0022-0000(92)90026-F. URL `http://dx.doi.org/10.1016/0022-0000(92)90026-F`.

[16] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*.

(MacGraw-Hill, New York, 1967).
[17] P. Odifreddi, *Classical Recursion Theory, Vol. 1.* (North-Holland, Amsterdam, 1989).
[18] P. Odifreddi, *Classical Recursion Theory, Vol. 2.* (North-Holland, Amsterdam, 1999).
[19] G. J. Chaitin, *Algorithmic Information Theory.* (Cambridge University Press, Cambridge, 1987).
[20] C. Calude, *Information and Randomness—An Algorithmic Perspective.* (Springer, Berlin, 1994).
[21] C. S. Calude and M. J. Dinneen. Exact approximations of omega numbers. URL `http://www.cs.auckland.ac.nz/CDMTCS//researchreports/293crismjd.pdf`. CDMTCS report series 293, (2006).
[22] E. Specker, Nicht konstruktiv beweisbare Sätze der Analysis, *The Journal of Smbolic Logic.* **14**, 145–158, (1949). Reprinted in [25, pp. 35–48]; English translation: *Theorems of Analysis which cannot be proven constructively.*
[23] E. Specker. Der Satz vom Maximum in der rekursiven Analysis. In ed. A. Heyting, *Constructivity in mathematics : proceedings of the colloquium held at Amsterdam, 1957*, pp. 254–265, Amsterdam, (1959). North-Holland Publishing Company. Reprinted in [25, pp. 148–159]; English translation: *Theorems of Analysis which cannot be proven constructively.*
[24] A. Adamatzky, *Collision-based computing.* (Springer, London, 2002).
[25] E. Specker, *Selecta.* (Birkhäuser Verlag, Basel, 1990).

Chapter 14

Binary Lambda Calculus and Combinatory Logic

John Tromp

CWI, Kruislaan 413, 1098 SJ Amsterdam, Netherlands;
john.tromp@gmail.com[1]

In the first part, we introduce binary representations of both lambda calculus and combinatory logic terms, and demonstrate their simplicity by providing very compact parser-interpreters for these binary languages. Along the way we also present new results on list representations, bracket abstraction, and fixpoint combinators. In the second part we review Algorithmic Information Theory, for which these interpreters provide a convenient vehicle. We demonstrate this with several concrete upper bounds on program-size complexity, including an elegant self-delimiting code for binary strings.

14.1. Introduction

The ability to represent programs as data and to map such data back to programs (known as reification and reflection [9]), is of both practical use in metaprogramming [14] as well as theoretical use in computability and logic [17]. It comes as no surprise that the pure lambda calculus, which represents both programs and data as functions, is well equipped to offer these features. Kleene [7] was the first to propose an encoding of lambda terms, mapping them to Gödel numbers, which can in turn be represented as so called Church numerals. Decoding such numbers is somewhat cumbersome, and not particularly efficient. In search of simpler constructions, various alternative encodings have been proposed using higher-order abstract syntax [8] combined with the standard lambda representation of signatures [11]. A particularly simple encoding was proposed by Mogensen [22], for which the term $\lambda m.m(\lambda x.x)(\lambda x.x)$ acts as a selfinterpreter. The prevalent data format, both in information theory and in practice, however, is

[1] no longer employed by CWI.

not numbers, or syntax trees, but bits. We propose binary encodings of both lambda and combinatory logic terms, and exhibit relatively simple and efficient interpreters (using the standard representation of bit-streams as lists of booleans).

This gives us a representation-neutral notion of the size of a term, measured in bits. More importantly, it provides a way to describe arbitrary data with, in a sense, the least number of bits possible. We review the notion of how a computer reading bits and outputting some result constitutes a description method, and how universal computer correspond to optimal description methods. We then pick specific universal computers based on our interpreters and prove several of the basic results of Algorithmic Information Theory with explicit constants.

14.2. Lambda Calculus

We only summarize the basics here. For a comprehensive treatment we refer the reader to the standard reference [18].

Assume a countably infinite set of *variables*

$$a, b, \ldots, x, y, z, x_0, x_1, \ldots$$

The set of lambda terms Λ is built up from variables using *abstraction*

$$(\lambda x.M)$$

and *application*

$$(M\ N),$$

where x is any variable and M, N are lambda terms. $(\lambda x.M)$ is the function that maps x to M, while $(M\ N)$ is the application of function M to argument N. We sometimes omit parentheses, understanding abstraction to associate to the right, and application to associate to the left, e.g. $\lambda x.\lambda y.x\ y\ x$ denotes $(\lambda x.(\lambda y.((x\ y)x)))$. We also join consecutive abstractions as in $\lambda x\ y.x\ y\ x$.

The free variables $FV(M)$ of a term M are those variables not bound by an enclosing abstraction. Λ^0 denotes the set of closed terms, i.e. with no free variables. The simplest closed term is the identity $\lambda x.x$.

We consider two terms *identical* if they only differ in the names of bound variables, and denote this with $\equiv$, e.g. $\lambda y.y\ x \equiv \lambda z.z\ x$. The essence of λ calculus is embodied in the β-conversion rule which equates

$$(\lambda x.M)N = M[x := N],$$

where $M[x := N]$ denotes the result of substituting N for all free occurrences of x in M (taking care to avoid variable capture by renaming bound variables in M if necessary). For example,

$$(\lambda x\ y.y\ x)y \equiv (\lambda x.(\lambda z.z\ x))y \equiv (\lambda x\ z.z\ x)y = \lambda z.z\ y.$$

A term with no β-redex, that is, no subterm of the form $(\lambda x.M)N$, is said to be in *normal form.* Terms may be viewed as denoting computations of which β-reductions form the steps, and which may halt with a normal form as the end result.

14.2.1. *Some useful lambda terms*

Define (for any $M, P, Q, \ldots, R$)

$$\begin{aligned}
\mathbf{I} &\equiv \lambda x.x \\
\mathbf{true} &\equiv \lambda x\ y.x \\
\mathbf{nil} \equiv \mathbf{false} &\equiv \lambda x\ y.y \\
\langle P, Q, \ldots, R\rangle &\equiv \lambda z.z\ P\ Q \ldots\ R \\
M[0] &\equiv M\ \mathbf{true} \\
M[i+1] &\equiv (M\ \mathbf{false})[i] \\
\mathbf{Y} &\equiv \lambda f.((\lambda x.x\ x)(\lambda x.f\ (x\ x))) \\
\mathbf{\Omega} &\equiv (\lambda x.x\ x)(\lambda x.x\ x)
\end{aligned}$$

Note that

$$\mathbf{true}\ P\ Q = (\lambda x\ y.x)\ P\ Q = x[x := P] = P$$

$$\mathbf{false}\ P\ Q = (\lambda x\ y.y)\ P\ Q = y[y := Q] = Q,$$

justifying the use of these terms as representing the booleans.

A pair of terms like P and Q is represented by $\langle P, Q\rangle$, which allows one to retrieve its parts by applying $\langle\mathbf{true}\rangle$ or $\langle\mathbf{false}\rangle$:

$$\langle\mathbf{true}\rangle\langle P, Q\rangle = \langle P, Q\rangle\ \mathbf{true} = \mathbf{true}\ P\ Q = P$$

$$\langle\mathbf{false}\rangle\langle P, Q\rangle = \langle P, Q\rangle\ \mathbf{false} = \mathbf{false}\ P\ Q = Q.$$

Repeated pairing is the standard way of representing a sequence of terms:

$$\langle P, \langle Q, \langle R, \ldots\rangle\rangle\rangle.$$

A sequence is thus represented by pairing its first element with its *tail*—the sequence of remaining elements. The i'th element of a sequence M may be selected as $M[i]$. To wit:

$$\langle P, Q\rangle[0] = \mathbf{true}\ P\ Q = P,$$

$$\langle P, Q\rangle[i+1] \equiv (\langle P, Q\rangle\ \mathbf{false})[i] = Q[i].$$

The empty sequence, for lack of a first element, cannot be represented by any pairing, and is instead represented by **nil**. A finite sequence $P, Q, \ldots, R$ can thus be represented as $\langle P, \langle Q, \langle \ldots, \langle R, \mathbf{nil}\rangle \ldots\rangle\rangle\rangle$.

Our choice of **nil** allows for the processing of a possible empty list s with the expression

$$s\ M\ N,$$

which for $s \equiv \mathbf{nil}$ reduces to N, and for $s \equiv \langle P, Q\rangle$ reduces to $M\ P\ Q\ N$]. In contrast, Barendregt [13] chose **I** to represent the empty list, which requires a much more complicated list processing expression like like $s\ (\lambda a\ b\ c.c\ a\ b)\ MXN$, which for $s = \mathbf{nil}$ reduces to $N\ M\ X$, and for $s \equiv \langle P, Q\rangle$ reduces to $M\ P\ Q\ X\ N$].

Y is the *fixpoint* operator, that satisfies

$$\mathbf{Y}f = (\lambda x.f\ (x\ x))(\lambda x.f\ (x\ x)) = f\ (\mathbf{Y}\ f).$$

This allows one to transform a recursive definition $f = \ldots f \ldots$ into $f = \mathbf{Y}(\lambda f.(\ldots f \ldots))$, which behaves exactly as desired.

Ω is the prime example of a term with no normal form, the equivalence of an infinite loop.

14.2.2. *Binary strings*

Binary strings are naturally represented by boolean sequences, where **true** represents 0 and **false** represents 1.

Definition 14.1. For a binary string s and lambda term M, $(s : M)$ denotes the list of booleans corresponding to s, terminated with M. Thus, $(s : \mathbf{nil})$ is the standard representation of string s.

For example, $(011 : \mathbf{nil}) \equiv \langle \mathbf{true}, \langle \mathbf{false}, \langle \mathbf{false}, \mathbf{nil}\rangle\rangle\rangle$ represents the string 011. We represent an unterminated string, such as part of an input stream, as an open term $(s : z)$, where the free variable z represents the remainder of input.

14.2.3. *de Bruijn notation*

de Bruijn [12] proposed an alternative notation for closed lambda terms using natural numbers rather than variable names. Abstraction is simply written λM while the variable bound by the n'th enclosing λ is written as the index n. In this notation, $\lambda x\ y\ z.z\ x\ y \equiv \lambda\ \lambda\ \lambda\ 0\ 2\ 1$. It thus provides a canonical notation for all identical terms. Beta-conversion in this notation avoids variable capture, but instead requires *shifting* the indices, i.e. adjusting them to account for changes in the lambda nesting structure. Since variable/index exchanges don't affect each other, it's possible to mix both forms of notation, as we'll do later.

14.2.4. *Binary Lambda Calculus*

Definition 14.2. The code for a term in de Bruijn notation is defined inductively as follows:

$$\begin{aligned}\widehat{n} &\equiv 1^{n+1}0\\ \widehat{\lambda M} &\equiv 00\widehat{M}\\ \widehat{MN} &\equiv 01\widehat{M}\ \widehat{N}\end{aligned}$$

We call $|\widehat{M}|$ the *size* of M.

For example $\widehat{\mathbf{I}} \equiv 0010$, $\widehat{\mathbf{false}} \equiv 000010$, $\widehat{\mathbf{true}} \equiv 0000110$ and $\widehat{\lambda x.x\ x} \equiv 00011010$, $\widehat{\lambda x.\mathbf{false}} \equiv 00000010$, of sizes 4,6,7,8 and 8 bits respectively, are the 5 smallest closed terms.

The main result of this paper is the following

Theorem 14.1. *There is a self-interpreter* $\mathbf{E}$ *of size 210 (which happens to be the product of the smallest four primes), such that for every closed term* M *and terms* C, N *we have*

$$\mathbf{E}\ C\ (\widehat{M} : N) = C\ (\lambda z.M)\ N$$

The interpreter works in continuation passing style [15]. Given a continuation and a bitstream containing an encoded term, it returns the continuation applied to the abstracted decoded term and the remainder of the stream. The reason for the abstraction becomes evident in the proof.

The theorem is a special case of a stronger one that applies to arbitrary de Bruijn terms. Consider a de Bruijn term M in which an index n occurs

at a depth of $i \leq n$ nested lambda's. E.g., in $M \equiv \lambda 3$, the index 3 occurs at depth 1. This index is like a free variable in that it is not bound within M. The interpreter (being a closed term) applied to other closed terms, cannot produce anything but a closed term. So it cannot possibly reproduce M. Instead, it produces terms that expect a list of bindings for free indices. These take the form $M^{z[]}$, which is defined as the result of replacing every free index in M, say n at depth $i \leq n$, by $z[n-i]$. For example, $(\lambda 3)^{z[]} = \lambda z[3-1] = \lambda(z\ \mathbf{false}\ \mathbf{false}\ \mathbf{true})$, selecting binding number 2 from binding list z.

The following claim (using mixed notation) will be needed later.

Claim 14.1. *For any de Bruijn term M, we have $(\lambda M)^{z[]} = \lambda y.M^{\langle y,z\rangle[]}$*

Proof. A free index n at depth $i \leq n$ in M, gets replaced by $\langle y,z\rangle[n-i]$ on the right. If $i < n$ then n is also free in λM at depth $i+1$ and gets replaced by $z[n-i-1] = \langle y,z\rangle[n-i]$. If $i = n$ then n is bound by the front λ, while $\langle y,z\rangle[n-i] = \langle y,z\rangle[0] = y$. □

To prove Theorem 14.1 it suffices to prove the more general:

Theorem 14.2. *There is a self-interpreter* $\mathbf{E}$ *of size 210, such that for all terms M, C, N we have*

$$\mathbf{E}\ C\ (\widehat{M} : N) = C\ (\lambda z.M^{z[]})\ N$$

Proof. We take

$$\begin{aligned}
\mathbf{E} &\equiv \mathbf{Y}\ (\lambda e\ c\ s.s\ (\lambda a\ t.t\ (\lambda b.a\ \mathbf{E}_0\ \mathbf{E}_1))) \\
\mathbf{E}_0 &\equiv e\ (\lambda x.b\ (c\ (\lambda z\ y.x\ \langle y,z\rangle))(e\ (\lambda y.c\ (\lambda z.x\ z\ (y\ z))))) \\
\mathbf{E}_1 &\equiv (b\ (c\ (\lambda z.z\ b))(\lambda s.e\ (\lambda x.c\ (\lambda z.x\ (z\ b)))\ t))
\end{aligned}$$

of size 217 and note that the beta reduction from $\mathbf{Y}\ M$ to $(\lambda x.x\ x)(\lambda x.M\ (x\ x))$ saves 7 bits.

Recall from the discussion of $\mathbf{Y}$ that the above is a transformed recursive definition where e will take the value of $\mathbf{E}$.

Intuitively, $\mathbf{E}$ works as follows. Given a continuation c and sequence s, it extracts the leading bit a and tail t of s, extracts the next bit b, and selects $\mathbf{E}_0$ to deal with $a = \mathbf{true}$ (abstraction or application), or $\mathbf{E}_1$ to deal with $a = \mathbf{false}$ (an index).

$\mathbf{E}_0$ calls $\mathbf{E}$ recursively, extracting a decoded term x. In case $b = \mathbf{true}$ (abstraction), it prepends a new variable y to bindings list z, and returns

the continuation applied to the decoded term provided with the new bindings. In case $b = \mathbf{false}$ (application), it calls $\mathbf{E}$ recursively again, extracting another decoded term y, and returns the continuation applied to the application of the decoded terms provided with shared bindings.

$\mathbf{E}_1$, in case $b = \mathbf{true}$, decodes to the 0 binding selector. In case $b = \mathbf{false}$, it calls $\mathbf{E}$ recursively on t (coding for an index one less) to extract a binding selector x, which is provided with the tail $z\ b$ of the binding list to obtain the correct selector.

We continue with the formal proof, using induction on M.

Consider first the case where $M = 0$. Then

$$\begin{aligned}
\mathbf{E}\ C\ (\widehat{M} : N) &= \mathbf{E}\ C\ (10 : N) \\
&= \langle \mathbf{false}, \langle \mathbf{true}, N\rangle\rangle\ (\lambda a\ t.t\ (\lambda b.a\ \mathbf{E}_0\ \mathbf{E}_1)) \\
&= \langle \mathbf{true}, N\rangle\ (\lambda b.\mathbf{false}\ \mathbf{E}_0\ \mathbf{E}_1) \\
&= (\mathbf{E}_1\ N)[b := \mathbf{true}] \\
&= C\ (\lambda z.z\ \mathbf{true})\ N,
\end{aligned}$$

as required. Next consider the case where $M = n + 1$. Then, by induction,

$$\begin{aligned}
\mathbf{E}\ C\ (\widehat{M} : N) &= \mathbf{E}\ C\ (1^{n+2}0 : N) \\
&= \langle \mathbf{false}, \langle \mathbf{false}, (1^n 0 : N)\rangle\rangle\ (\lambda a\ t.t\ (\lambda b.a\ \mathbf{E}_0\ \mathbf{E}_1)) \\
&= (\lambda s.e\ (\lambda x.C\ (\lambda z.x\ (z\ \mathbf{false})))(1^{n+1}0 : N))(1^n 0 : N) \\
&= \mathbf{E}\ (\lambda x.C\ (\lambda z.x\ (z\ \mathbf{false})))\ (\widehat{n} : N) \\
&= (\lambda x.C\ (\lambda z.x\ (z\ \mathbf{false})))\ (\lambda z.n^{z[]})\ N \\
&= C\ (\lambda z.n^{(z\ \mathbf{false})[]})\ N \\
&= C\ (\lambda z.(z\ \mathbf{false})[n]))\ N \\
&= C\ (\lambda z.z[n+1]))\ N \\
&= C\ (\lambda z.(n+1)^{z[]})\ N,
\end{aligned}$$

as required. Next consider the case $M = \lambda M'$. Then, by induction and claim 14.1,

$$\begin{aligned}
\mathbf{E}\ C\ ((\widehat{\lambda M'} : N)) &= \mathbf{E}\ C\ (00\widehat{M'} : N) \\
&= \langle \mathbf{true}, \langle \mathbf{true}, (\widehat{M'} : N)\rangle\rangle\ (\lambda a\ t.t\ (\lambda b.a\ \mathbf{E}_0\ \mathbf{E}_1)) \\
&= e\ (\lambda x.(C\ (\lambda z\ y.x\langle y, z\rangle)))\ (\widehat{M'} : N) \\
&= (\lambda x.(C\ (\lambda z\ y.x\ \langle y, z\rangle)))(\lambda z.M'^{z[]})\ N
\end{aligned}$$

$$
\begin{aligned}
&= C\ (\lambda z\ y.(\lambda z.M'^{z[]})\ \langle y,z\rangle)\ N\\
&= C\ (\lambda z.(\lambda y.M'^{\langle y,z\rangle[]}))\ N\\
&= C\ (\lambda z.(\lambda M')^{z[]}))\ N,
\end{aligned}
$$

as required. Finally consider the case $M = M'\ M''$. Then, by induction,

$$
\begin{aligned}
\mathbf{E}\ C\ (\widehat{M'\ M''} : N) &= \mathbf{E}\ C\ (01\widehat{M'}\ \widehat{M''} : N)\\
&= \langle \mathbf{true}, \langle \mathbf{false}, (\widehat{M'}\ \widehat{M''} : N)\rangle\rangle(\lambda a\ t.t\ (\lambda b.a\ \mathbf{E}_0\ \mathbf{E}_1))\\
&= e\ (\lambda x.(e\ (\lambda y.C\ (\lambda z.x\ z\ (y\ z)))))\ (\widehat{M'}\ \widehat{M''} : N)\\
&= (\lambda x.(e\ (\lambda y.C\ (\lambda z.x\ z\ (y\ z)))))(\lambda z.M'^{z[]})\ (\widehat{M''} : N)\\
&= e\ (\lambda y.C\ (\lambda z.(\lambda z.M'^{z[]})\ z\ (y\ z)))(\widehat{M''} : N)\\
&= (\lambda y.C\ (\lambda z.M'^{z[]}\ (y\ z)))(\lambda z.M''^{z[]})\ N\\
&= C\ (\lambda z.M'^{z[]}\ M''^{z[]})\ N\\
&= C\ (\lambda z.(M'\ M'')^{z[]})\ N,
\end{aligned}
$$

as required. This completes the proof of Theorem 14.1. □

We conjecture that E is the smallest self-interpreter for any binary representation of lambda calculus.

14.3. Combinatory Logic

Combinatory Logic (CL) is the equational theory of *combinators*—terms built up, using application only, from the two constants **K** and **S**, which satisfy

$$
\begin{aligned}
\mathbf{S}\ M\ N\ L &= M\ L\ (N\ L)\\
\mathbf{K}\ M\ N &= M
\end{aligned}
$$

CL may be viewed as a subset of lambda calculus, in which $\mathbf{K} \equiv \lambda x\ y.x$, $\mathbf{S} \equiv \lambda x\ y\ z.x\ z\ (y\ z)$, and where the beta conversion rule can only be applied groupwise, either for an **S** with 3 arguments, or for a **K** with 2 arguments. Still, the theories are largely the same, becoming equivalent in the presence of the rule of extensionality (which says $M = M'$ if $M\ N = M'\ N$ for all terms N).

A process known as *bracket abstraction* allows for the translation of any lambda term to a *combination*—a CL term containing variables in addition

to **K** and **S**. It is based on the following identities, which are easily verified:

$$\begin{aligned}\lambda x.x &= \mathbf{I} = \mathbf{S}\ \mathbf{K}\ \mathbf{K}\\ \lambda x.M &= \mathbf{K}\ M \qquad (x \text{ not free in } M)\\ \lambda x.M\ N &= \mathbf{S}\ (\lambda x.M)\ (\lambda x.N)\end{aligned}$$

λ's can thus be successively eliminated, e.g.:

$$\begin{aligned}\lambda x\ y.y\ x &\equiv \lambda x\ (\lambda y.y\ x)\\ &= \lambda x\ (\mathbf{S}\ \mathbf{I}(\mathbf{K}\ x))\\ &= \mathbf{S}\ (\mathbf{K}\ (\mathbf{S}\ \mathbf{I}))(\mathbf{S}\ (\mathbf{K}\ \mathbf{K})\ \mathbf{I}),\end{aligned}$$

where **I** is considered a shorthand for **S K K**.

Bracket abstraction is an operation λ^0 on combinations M with respect to a variable x, such that the resulting combination contains no occurrence of x and behaves as $\lambda x.M$:

$$\begin{aligned}\lambda^0 x.\ x &\equiv \mathbf{I}\\ \lambda^0 x.\ M &\equiv \mathbf{K}\ M \qquad (x \notin M)\\ \lambda^0 x.\ (M\ N) &\equiv \mathbf{S}\ (\lambda^0 x.\ M)\ (\lambda^0 x.\ N)\end{aligned}$$

14.3.1. *Binary Combinatory Logic*

Combinators have a wonderfully simple encoding as binary strings: encode **S** as 00, **K** as 01, and application as 1.

Definition 14.3. We define the encoding $\widetilde{C}$ of a combinator C as

$$\begin{aligned}\widetilde{\mathbf{S}} &\equiv 00\\ \widetilde{\mathbf{K}} &\equiv 01\\ \widetilde{C\ D} &\equiv 1\ \widetilde{C}\ \widetilde{D}\end{aligned}$$

Again we call $|\widetilde{C}|$ the *size* of combinator C.

For instance, the combinator $\mathbf{S}(\mathbf{KSS}) \equiv (\mathbf{S}((\mathbf{KS})\mathbf{S}))$ is encoded as 10011010000. The size of a combinator with n **K**/**S**'s, which necessarily has $n-1$ applications, is thus $2n+n-1 = 3n-1$.

For such a simple language we expect a similarly simple interpreter.

Theorem 14.3. *There is a cross-interpreter* **F** *of size 124, such that for every combinator M and terms C, N we have*

$$\mathbf{F}\ C\ (\widetilde{M} : N) = C\ M\ N$$

Proof. We take

$$\begin{aligned}\mathbf{F} &\equiv \mathbf{Y}\ (\lambda e\ c\ s.s(\lambda a.a\ \mathbf{F}_0\ \mathbf{F}_1))\\ \mathbf{F}_0 &\equiv \lambda t.t\ (\lambda b.c\ (b\ \mathbf{S}\ \mathbf{K}))\\ \mathbf{F}_1 &\equiv e\ (\lambda x.e\ (\lambda y.(c\ x\ y)))\end{aligned}$$

of size 131 and note that a toplevel beta reduction saves 7 bits in size.

Given a continuation c and sequence s, it extracts the leading bit a of s, and tail t extracts the next bit b, and selects $\mathbf{F}_0$ to deal with $a =$ **true** (**S** or **K**), or $\mathbf{F}_1$ to deal with $a =$ **false** (application). Verification is straightforward and left as an exercise to the reader. □

We conjecture F to be the smallest interpreter for any binary representation of CL. The next section considers translations of F which yield a self-interpreter of CL.

14.3.2. *Improved bracket abstraction*

The basic form of bracket abstraction is not particularly efficient. Applied to **F**, it produces a combinator of size 536.

A better version is λ^1, which uses the additional rule

$$\lambda^1 x.\ (M\ x) \equiv M \quad (x \notin M)$$

whenever possible. Now the size of **F** as a combinator is only 281, just over half as big.

Turner [23] noticed that repeated use of bracket abstraction can lead to a quadratic expansion on terms such as

$$\mathbf{X} \equiv \lambda a\ b\ \ldots\ z.(a\ b\ \ldots\ z)\ (a\ b\ \ldots\ z),$$

and proposed new combinators to avoid such behaviour. We propose to achieve a similar effect with the following set of 9 rules in decreasing order

of applicability:

$$\begin{aligned}
\lambda^2 x.\ (\mathbf{S}\ \mathbf{K}\ M) &\equiv \mathbf{S}\ \mathbf{K} \quad \text{(for all } M\text{)} \\
\lambda^2 x.\ M &\equiv \mathbf{K}\ M \quad (x \notin M) \\
\lambda^2 x.\ x &\equiv \mathbf{I} \\
\lambda^2 x.\ (M\ x) &\equiv M \quad (x \notin M) \\
\lambda^2 x.\ (x\ M\ x) &\equiv \lambda^2 x.\ (\mathbf{S}\ \mathbf{S}\ \mathbf{K}\ x\ M) \\
\lambda^2 x.\ (M\ (N\ L)) &\equiv \lambda^2 x.\ (\mathbf{S}\ (\lambda^2 x.\ M)\ N\ L) \quad (M, N \text{ combinators}) \\
\lambda^2 x.\ ((M\ N)\ L) &\equiv \lambda^2 x.\ (\mathbf{S}\ M\ (\lambda^2 x.\ L)\ N) \quad (M, L \text{ combinators}) \\
\lambda^2 x.\ ((M\ L)\ (N\ L)) &\equiv \lambda^2 x.\ (\mathbf{S}\ M\ N\ L) \quad (M, N \text{ combinators}) \\
\lambda^2 x.\ (M\ N) &\equiv \mathbf{S}\ (\lambda^0 x.\ M)\ (\lambda^0 x.\ N)
\end{aligned}$$

The first rule exploits the fact that $\mathbf{S}\ \mathbf{K}\ M$ behaves as identity, whether M equals $\mathbf{K}, x$ or anything else. The fifth rule avoids introduction of two $\mathbf{I}$*s*. The sixth rule prevents occurrences of x in L from becoming too deeply nested, while the seventh does the same for occurrences of x in N. The eighth rule abstracts an entire expression L to avoid duplication. The operation $\lambda^2 x.\ M$ for combinators M will normally evaluate to $\mathbf{K}\ M$, but takes advantage of the first rule by considering any $\mathbf{S}\ \mathbf{K}\ M$ a combinator. Where λ^1 gives an $\mathbf{X}$ combinator of size 2030, λ^2 brings this down to 374 bits.

For $\mathbf{F}$ the improvement is more modest, to 275 bits. For further improvements we turn our attention to the unavoidable fixpoint operator.

$\mathbf{Y}$, due to Curry, is of minimal size in the λ calculus. At 25 bits, it's 5 bits shorter than Turing's alternative fixpoint operator

$$\mathbf{Y}' \equiv (\lambda z.z\ z)(\lambda z.\lambda f.f\ (z\ z\ f)).$$

But these translate to combinators of size 65 and 59 bits respectively.

In comparison, the fixpoint operator

$$\mathbf{Y}'' \equiv (\lambda x\ y.x\ y\ x)(\lambda y\ x.y(x\ y\ x))$$

translates to combinator

$$\mathbf{S}\ \mathbf{S}\ \mathbf{K}\ (\mathbf{S}\ (\mathbf{K}\ (\mathbf{S}\ \mathbf{S}\ (\mathbf{S}\ (\mathbf{S}\ \mathbf{S}\ \mathbf{K}))))\ \mathbf{K})$$

of size 35, the smallest possible fixpoint combinator as verified by exhaustive search by computer.

(The situation is similar for $\mathbf{\Omega}$ which yields a combinator of size 41, while $\mathbf{S}\ \mathbf{S}\ \mathbf{K}\ (\mathbf{S}\ (\mathbf{S}\ \mathbf{S}\ \mathbf{K}))$, of size 20, is the smallest *unsolvable* combinator—the equivalent of an undefined result, see Barendregt [18]).

Using $\mathbf{Y}''$ instead of $\mathbf{Y}$ gives us the following

Theorem 14.4. *There is a self-interpreter* $\mathbf{F}$ *for Combinatory Logic of size 263.*

Comparing theorems 14.3 and 14.4, we conclude that λ-calculus is a much more concise language than CL. Whereas in binary λ-calculus, an abstraction takes only 2 bits plus $i+1$ bits for every occurrence of the variable at depth i, in binary CL the corresponding bracket abstraction typically introduces at least one, and often several $\mathbf{S}$'s and $\mathbf{K}$'s (2 bits each) per level of depth per variable occurrence.

14.4. Program Size Complexity

Intuitively, the amount of information in an object is the size of the shortest program that outputs the object. The first billion digits of π for example, contain little information, since they can be calculated by a program of a few lines only. Although information content may seem to be highly dependent on choice of programming language, the notion is actually invariant up to an additive constant.

The theory of program size complexity, which has become known as *Algorithmic Information Theory* or *Kolmogorov complexity* after one of its founding fathers, has found fruitful application in many fields such as combinatorics, algorithm analysis, machine learning, machine models, and logic.

In this section we propose a concrete definition of complexity that is (arguably) as simple as possible, by turning the above interpreters into a 'universal computer'.

Intuitively, a computer is any device that can read bits from an input stream, perform computations, and (possibly) output a result. Thus, a computer is a method of description in the sense that the string of bits read from the input describes the result. A universal computer is one that can emulate the behaviour of any other computer when provided with its description. Our objective is to define, concretely, for any object x, a measure of complexity of description $C(x)$ that shall be the length of its shortest description. This requires fixing a description method, i.e. a computer. By choosing a universal computer, we achieve invariance: the complexity of objects is at most a constant greater than under any other description method.

Various types of computers have been considered in the past as description methods.

Turing machines are an obvious choice, but turn out to be less than ideal: The operating logic of a Turing machine—its *finite control*—is of an irregular nature, having no straightforward encoding into a bitstring. This makes construction of a universal Turing machine that has to parse and interpret a finite control description quite challenging. Roger Penrose takes up this challenge in his book [1], at the end of Chapter 2, resulting in a universal Turing machine whose own encoding is an impressive 5495 bits in size, over 26 times that of **E**.

The ominously named language 'Brainfuck' which advertises itself as "An Eight-Instruction Turing-Complete Programming Language" [21], can be considered a streamlined form of Turing machine. Indeed, Oleg Mazonka and Daniel B. Cristofani [16] managed to write a very clever BF self-interpreter of only 423 instructions, which translates to $423 \cdot \log(8) = 1269$ bits (the alphabet used is actually ASCII at 7 or 8 bits per symbol, but the interpreter could be redesigned to use 3-bit symbols and an alternative program delimiter).

In [5], Levin stresses the importance of a (descriptional complexity) measure, which, when compared with other natural measures, yields small constants, of at most a few hundred bits. His approach is based on *constructive objects* (c.o.'s) which are functions from and to lower ranked c.o.'s. Levin stops short of exhibiting a specific universal computer though, and the abstract, almost topological, nature of algorithms in the model complicates a study of the constants achievable.

Gregory Chaitin [2] paraphrases John McCarthy about his invention of LISP, as "This is a better universal Turing machine. Let's do recursive function theory that way!" Later, Chaitin continues with "So I've done that using LISP because LISP is simple enough, LISP is in the intersection between theoretical and practical programming. Lambda calculus is even simpler and more elegant than LISP, but it's unusable. Pure lambda calculus with combinators S and K, it's beautifully elegant, but you can't really run programs that way, they're too slow."

There is however nothing intrinsic to λ calculus or CL that is slow; only such choices as Church numerals for arithmetic can be said to be slow, but one is free to do arithmetic in binary rather than in unary. Frandsen and Sturtivant [10] amply demonstrate the efficiency of λ calculus with a linear time implementation of k-tree Turing Machines. Clear semantics should be a primary concern, and Lisp is somewhat lacking in this regard [4]. This paper thus develops the approach suggested but discarded by Chaitin.

14.4.1. *Functional Complexity*

By providing the appropriate continuations to the interpreters that we constructed, they become universal computers describing functional terms modulo equality. Indeed, for

$$\mathbf{U} \equiv \mathbf{E}\ \langle \mathbf{\Omega} \rangle$$
$$\mathbf{U}' \equiv \mathbf{F}\ \mathbf{I}$$

of sizes $|\widehat{\mathbf{U}}| = 236$ and $|\widetilde{\mathbf{U}'}| = 272$, Theorems 14.1 and 14.3 give

$$\mathbf{U}\ (\widehat{M} : N) = M\ N$$
$$\mathbf{U}'\ (\widetilde{M} : N) = M\ N$$

for every closed λ-term or combinator M and arbitrary N, immediately establishing their universality.

The universal computers essentially define new binary languages, which we may call *universal binary lambda calculus* and *universal combinatory logic*, whose programs comprise two parts. The first part is a program in one of the original binary languages, while the second part is all the binary data that is consumed when the first part is interpreted. It is precisely this ability to embed arbitrary binary data in a program that allows for universality.

Note that by Theorem 14.2, the continuation $\langle \mathbf{\Omega} \rangle$ in U results in a term $M^{\mathbf{\Omega}[]}$. For closed M, this term is identical to M, but in case M is not closed, a free index n at λ-depth n is now bound to $\mathbf{\Omega}[n-n]$, meaning that any attempt to apply free indices diverges. Thus the universal computer essentially forces programs to be closed terms.

We can now define the complexity of a term x, which comes in three flavors. In the *simple* version, programs are terminated with $N = \mathbf{nil}$ and the result must equal x. In the *prefix* version, programs are not terminated, and the result must equal the pair of x and the remainder of the input. In both cases the complexity is conditional on zero or more terms y_i.

Definition 14.4.

$$KS(x|y_1, \ldots, y_k) = \min\{l(p) \mid \mathbf{U}\ (p : \mathbf{nil})\ y_1\ \ldots\ y_k = x\}$$
$$KP(x|y_1, \ldots, y_k) = \min\{l(p) \mid \mathbf{U}\ (p : z)\ y_1\ \ldots\ y_k = \langle x, z \rangle\}$$

In the special case of $k = 0$ we obtain the unconditional complexities $KS(x)$ and $KP(x)$.

Finally, for a binary string s, we can define its *monotone* complexity as

$$KM(s|y_1,\ldots,y_k) = \min\{l(p) \mid \exists M : \mathbf{U}\ (p : \mathbf{\Omega})\ y_1\ \ldots\ y_k = (s : M)\}.$$

In this version, we consider the partial outputs produced by increasingly longer prefixes of the input, and the complexity of s is the shortest program that causes the output to have prefix s.

14.4.2. *Monadic IO*

The reason for preserving the remainder of input in the prefix casse is to facilitate the processing of concatenated descriptions, in the style of monadic IO [19]. Although a pure functional language like λ calculus cannot define functions with side effects, as traditionally used to implement IO, it can express an abstract data type representing IO actions; the IO monad. In general, a monad consists of a type constructor and two functions, *return* and *bind* (also written `>>=` in infix notation) which need to satisfy certain axioms [19]. IO actions can be seen as functions operating on the whole state of the world, and returning a new state of the world. Type restrictions ensure that IO actions can be combined only through the bind function, which according to the axioms, enforces a sequential composition in which the world is single-threaded. Thus, the state of the world is never duplicated or lost. In our case, the world of the universal machine consists of only the input stream. The only IO primitive needed is **readBit**, which maps the world onto a pair of the bit read and the new world. But a list is exactly that; a pair of the first element and the remainder. So **readBit** is simply the identity function! The **return** function, applied to some x, should map the world onto the pair of x and the unchanged world, so it is defined by $\mathbf{return} \equiv \lambda x\ y.\langle x, y\rangle$. Finally, the bind function, given an action x and a function f, should subject the world y to action x (producing some $\langle a, y'\rangle$) followed by action fa, which is defined by $\mathbf{bind} \equiv \lambda x\ f\ y.x\ y\ f$ (note that $\langle a, y'\rangle f = f\ a\ y'$) One may readily verify that these definitions satisfy the monad axioms. Thus, we can wite programs for U either by processing the input stream explicitly, or by writing the program in monadic style. The latter can be done in the pure functional language 'Haskell' [20], which is essentially typed lambda calculus with a lot of syntactic sugar.

14.4.3. *An Invariance Theorem*

The following theorem is the first concrete instance of the Invariance Theorem, 2.1.1 in Li&Vitányi [6].

Theorem 14.5. *Define* $KS'(x|y_1,\ldots,y_k)$ *and* $KP'(x|y_1,\ldots,y_k)$ *analogous to Definition 14.4 in terms of* $\mathbf{U}'$. *Then* $KS(x) \leq KS'(x) + 130$ *and* $KP(x) \leq KP'(x) + 130$.

The proof is immediate from Theorem 14.3 by using $\widehat{\mathbf{U}'}$ of length 130 as prefix to any program for $\mathbf{U}'$. We state without proof that a redesigned $\mathbf{U}$ translates to a combinator of size 617, which thus forms an upper bound in the other direction.

Now that complexity is defined for as rich a class of objects as terms (modulo equality), it is easy to extend it to other classes of objects by mapping them into λ terms.

For binary strings, this means mapping string s onto the term $(s : \mathbf{nil})$. And for a tuple of binary strings $s_0,\ldots,s_k$, we take $\langle(s_0 : \mathbf{nil}),\ldots,(s_k : \mathbf{nil})\rangle$.

We next look at numbers in more detail, revealing a link with self-delimiting strings.

14.4.4. *Numbers and Strings*

Consider the following correspondence between natural numbers and binary strings

$$\begin{array}{r} n \in \mathbb{N} : 0\ 1\ 2\ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ \ 8\ \ \ 9\ \ \ldots \\ x \in \{0,1\}^* : \epsilon\ 0\ 1\ 00\ 10\ 01\ 11\ 000\ 100\ 010\ \ldots \end{array}$$

which can be described in several ways. The number n corresponds to *the reverse* of

- the n-th binary string in lexicographic order
- the string obtained by stripping the leading 1 from $(n+1)_2$, the binary representation of $n+1$
- the string obtained by renaming digits in $n_{\{1,2\}}$, the base 2 positional system using digits $\{1,2\}$

There are two reasons for taking the reverse above. Bits are numbered from right to left in a positional system where position i carries weight 2^i, while our list representation is inherently left to right. Second, almost all

operations on numbers process the bits from least to most significant, so the least significant bits should come first in the list.

The $\{1,2\}$-notation is interesting in that it not only avoids the problem of leading zeroes but actually forces a unique representation:

$$\begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \dots \\ \epsilon & 1 & 2 & 11 & 21 & 12 & 22 & 111 & 211 & 121 & \dots \end{array}$$

Note that if n corresponds to the string $x = x_{l-1} \dots x_0 = X_2$, then according to equivalent $\{1,2\}$-notation,

$$n = \sum_{i=0}^{l-1} (x_i + 1) 2^i = \sum_{i=0}^{l-1} 2^i + \sum_{i=0}^{l-1} x_i 2^i = 2^l - 1 + X.$$

Hence, $n + 1 = 2^l + X$ which reconfirms the 2nd correspondence above.

14.4.5. *Prefix codes*

Another way to tie the natural numbers and binary strings together is the *binary natural tree* shown in Figure 14.1. It has the set of natural numbers as vertices, and the set of binary strings as edges, such that the 2^n length-n strings are the edges leading from vertex n. Edge w leads from vertex $|w|$ to $w + 1$, which in binary is $1w$.

Consider the concatenated edges on the path from 0 to n, which we'll denote by $p(n)$. The importance of the binary natural tree lies in the observation that the set of all $p(n)$ is almost *prefix-free*. In a prefix-free set, no string is a proper prefix of another, which is the same as saying that the strings in the set are self-delimiting. Prefix-free sets satisfy the *Kraft inequality*: $\sum_s 2^{-|s|} \leq 1$. We've already seen two important examples of prefix-free sets, namely the set of λ term encodings $\widehat{M}$ and the set of combinator encodings $\widetilde{M}$. To turn $p(n)$ into a prefix code, it suffices to prepend the depth of vertex n in the tree, i.e. the number of times we have to map n to $|n-1|$ before we get to ϵ. Denoting this depth as $l^*(n)$, we obtain the prefix code

$$\overline{n} = 1^{l^*(n)} 0 \; p(n),$$

or, equivalently,

$$\overline{0} = 0 \qquad \overline{n+1} = 1 \; \overline{l(n)} \; n.$$

This satisfies the following nice properties:

- prefix-free and complete: $\sum_{n \geq 0} 2^{-|\overline{n}|} = 1$.

- simple to encode and decode
- efficient in that for every k: $|\overline{n}| \leq l(n)+l(l(n))+\cdots+l^{k-1}(n)+O(l^k(n))$, where $l(s)$ denotes the length of a string s.

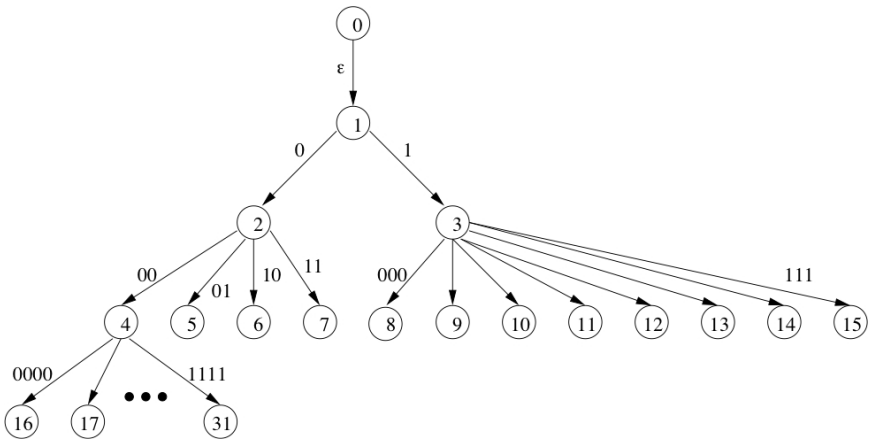


Figure 14.1. binary natural tree

Figure 14.2. codes on the unit interval; $\overline{0} = 0, \overline{1} = 10, \overline{2} = 110\ 0, \overline{3} = 110\ 1, \overline{4} = 1110\ 0\ 00, \overline{5} = 1110\ 0\ 10, \overline{6} = 1110\ 0\ 01, \overline{7} = 1110\ 0\ 11, \overline{8} = 1110\ 1\ 000, etc.$.

Figure 14.2 shows the codes as segments of the unit interval, where code x covers all the real numbers whose binary expansion starts as $0.x$.

14.5. Upper bounds on complexity

Having provided concrete definitions of all key ingredients of algorithmic information theory, it is time to prove some concrete results about the complexity of strings.

The simple complexity of a string is upper bounded by its length:

$$KS(x) \leq |\widehat{\mathbf{I}}| + l(x) = l(x) + 4$$

The prefix complexity of a string is upper bounded by the length of its

delimited version:

$$KP(x) \leq |\widehat{\mathbf{delimit}}| + l(\overline{x}) = l(\overline{x}) + 413.$$

where **delimit** is a straightforward translation of the following Haskell code into λ calculus:

```
delimit = do bit <- readBit
             if bit then return []
                    else do len <- delimit
                            n <- readbits len
                            return (inc n)
  where
  readbits [] = return []
  readbits len = do bit <- readBit
                    x <- readbits (dec len)
                    return (bit:x)
  dec [True] = []
  dec (True:rest) = False:(dec rest)
  dec (False:rest) = True:rest

  inc [] = [True]
  inc (True:rest) = False:rest
  inc (False:rest) = True:(inc rest)
```

The 'do' notation is syntactic sugar for the binding operator `>>=`, as exemplified by the following de-sugared version of **readbits**:

```
  readbits len = readBit >>= (\bit ->
                 readbits (dec len) >>= (\x ->
                 return (bit:x)))
```

The prefix complexity of a pair is upper bounded by the sum of individual prefix complexities, one of which is conditional on the shortest program of the other:

$$K(x, y) \leq K(x) + K(y|x^*) + 1876.$$

This is the easy side of the fundamental "Symmetry of information" theorem $K(x) - K(x|y^*) = K(y) - K(y|x^*) + O(1)$, which says that y contains as much information about x as x does about y.

Chaitin [3] proves the same theorem using a resource bounded evaluator, which in his version of LISP comes as a primitive called "try". His proof is embodied in the program gamma:

```
((’(lambda (loop) ((’(lambda (x*) ((’(lambda (x) ((’(lambda
(y) (cons x (cons y nil)))) (eval (cons (’(read-exp)) (cons
(cons ’(cons x* nil)) nil)))))) (car (cdr (try no-time-limit
(’(eval (read-exp))) x*)))))) (loop nil)))) (’(lambda (p)
(if(= success (car (try no-time-limit (’(eval (read-exp)))
p))) p (loop (append p (cons (read-bit) nil)))))))
```

of length 2872 bits.

We constructed an equivalent of "try" from scratch. The constant 1876 is the size of the term `pairup` defined below, containing a symbolic lambda calculus normal form reducer (which due to space restrictions is only sparsely commented):

```
-- identity
I = \x x

-- ‘bool x y’ represents ‘if bool then x else y’
true = \x\y x
false = \x\y y

-- allows for list processing as: list (\head\tail\x case-non-nil)
case-nil nil = false

-- unary number representation
zero = false
one = \s s zero
succ = \n\s s n
pred = \n n I

-- binary Lambda Calculus interpreter
intL = \cont\list list (\bit0\list1 list1 (\bit1 bit0
  (intL (\exp bit1 (cont (\args\arg exp (\z z arg args)))
                (intL (\exp2 cont (\args exp args (exp2 args))))))
  (bit1 (cont (\args args bit1))
       (\list2 intL (\var cont (\args var (args bit1))) list1))))
-- binary Lambda Calculus universal machine allowing open programs
uniL’ = intL (\x x x)

readvar = \cont\list list (\bit0 bit0
   (cont (\suff \z z bit0 suff)nil )
   (readvar (\pref\v cont (\suff \z z bit0 (pref suff)) (\s s v))))

-- binary Lambda Calculus parser
readc = \cont\list list (\bit0 bit0
  (\list1 list1 (\bit1 readc (\pref1\exp1 bit1
    (cont (\suff \z z bit0 (\z z bit1 (pref1 suff))) (\l\a\v l exp1))
    (readc (\pr2\exp2 cont (\suff \z z bit0 (\z z bit1 (pr1 (pr2 suff))))
(\l\a\v a exp1 exp2) )))))
  (readvar (\pref\var cont (\suff \z z bit0 (pref suff)) (\l\a\v v var))))
```

```
-- apply (\var (succ^i k) var var) to variables in lam at depth i
shift = \k\lam lam
  (\lam   \l\a\v l (shift (\s s k) lam))
  (\t1\t2 \l\a\v a (shift k t1) (shift k t2))
  (\var   \l\a\v v (k var var))
-- 'k var' will be one of succ, pred or identity

-- used in place of zero for variables to be substituted
eqnil = \z z I z

-- substitute e for free variable i (of the form k = succ^i eqnil) in term
subst = \e\k\term term
  (\lam   \l\a\v l (subst (shft (\z succ) e) (\s s k) lam))
  (\t1\t2 \l\a\v a (subst e k t1) (subst e k t2))
  (\var   k var (\x term) e)

shdec = shft (\z pred)
shinc = shft (\z succ)

whnfreduce = \term term
  (\body term)
  (\term1\term2 (\whnf1 (\nfterm whnf1
     (\body whnfreduce ((\shft shdec (subst (shinc term2) eqnil body)) shift))
     (\t1 nfterm)
     nfterm) (\x\l\a\v a whnf1 term2)) (whnfreduce term1))
  (\var term)

readbits = \k\list\fun k (\k1\tail list (\bit\list1
            fun bit (readbits k1 list1 fun tail)))
pair = \x\y\z z x y
symprebit = \x\y \l\a\v l (\l\a\v a (\l\a\v a (\l\a\v v zero)
      (\l\a\v l (\l\a\v l (\l\a\v v (x one zero))))) y)
symfalse = (\l\a\v l (\l\a\v l (\l\a\v v zero)))

-- minp returns the smallest prefix of at least k bits of data that prog
-- will 'read' from it to produce a pair <_,restofdata>
-- it does so by symbolically reducing prog applied to data' applied to
false
-- where data' is the first k bits of data terminated with de Bruijn
variable k
-- the solitary rdbits is a dummy; the reduction should never yield a
lambda minp = \prog\data\k (\rdbits
 whnfreduce (\l\a\v a (\l\a\v a
     prog (rdbits k data symprebit (\l\a\v v k))) symfalse)
   rdbits
   (\t1\t2 (minp prog data (\s s k)))
   (\v rdbits k data pair nil)) readbits

-- program list returns the program at the start of binary stream list
program = readc (\pref\prog\data pref (minp prog data zero))

-- pairup listpq returns <<x,y>,z> if the binary stream listpq
-- starts with a program p for x, followed by a program q for
-- computing y given p, followed by the remainder stream z
pairup = \listpq (\uni uni listpq
           (\x\listq uni listq (program listpq)
             (\y\list \z z (\z z x y) list))
         ) uniL'
```

Although seemingly much more complex, our program is actually shorter when measured in bits! Chaitin also offered a program of size only

2104 bits, at the cost of introducing yet another primitive into his language. Our program is 996 bits shorter than his first, and 228 bits shorter than his second.

14.6. Future Research

It would be nice to have an objective measure of the simplicity and expressiveness of a universal machine. Sizes of constants in fundamental theorems are an indication, but one that is all too easily abused. Perhaps diophantine equations can serve as a non-arbitrary language into which to express the computations underlying a proposed definition of algorithmic complexity, as Chaitin has demonstrated for relating the existence of infinitely many solutions to the random halting probability Ω. Speaking of Ω, our model provides a well-defined notion of halting as well, namely when $\mathbf{U}\ (p : z) = \langle M, z\rangle$ for any term M (we might as well allow M without normal form). Computing upper and lower bounds on the value of Ω_λ, as Chaitin did for his LISP-based Ω, and Calude et al. for various other languages, should be of interest as well. A big task remains in finding a good constant for the other direction of the 'Symmetry of Information' theorem, for which Chaitin has sketched a program. That constant is bigger by an order of magnitude, making its optimization an everlasting challenge.

14.7. Conclusion

The λ-calculus is a surprisingly versatile and concise language, in which not only standard programming constructs like bits, tests, recursion, pairs and lists, but also reflection, reification, and marshalling are readily defined, offering an elegant concrete foundation of algorithmic information theory.

An implementation of Lambda Calculus, Combinatory Logic, along with their binary and universal versions, written in Haskell, is available at Tromp's website [24].

Acknowledgements

I am greatly indebted to Paul Vitányi for fostering my research into concrete definitions of Kolmogorov complexity, and to Robert Solovay for illuminating discussions on my definitions and on the above symbolic reduction engine in particular, which not only revealed a bug but lead me to significant further reductions in size.

References

[1] R. Penrose, *The Emperor's New Mind*, Oxford University press, 1989.

[2] G. Chaitin, *An Invitation to Algorithmic Information Theory*, DMTCS'96 Proceedings, Springer Verlag, Singapore, 1997, pp. 1–23 (`http://www.cs.auckland.ac.nz/CDMTCS/chaitin/inv.html`).

[3] G. Chaitin, *Exploring Randomness*, Springer Verlag, 2001. (`http://www.cs.auckland.ac.nz/CDMTCS/chaitin/ait3.html`)

[4] R. Muller, *M-LISP: A representation-independent dialect of LISP with reduction semantics*, ACM Transactions on Programming Languages and Systems 14(4), 589–616, 1992.

[5] L. Levin, *On a Concrete Method of Assigning Complexity Measures*, Doklady Akademii nauk SSSR, vol. 18(3), pp. 727–731, 1977.

[6] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Graduate Texts in Computer Science, second edition, Springer-Verlag, New York, 1997.

[7] S.C. Kleene, *Lambda-Definability and Recursiveness*, Duke Mathematical Journal, 2, 340–353, 1936.

[8] Frank Pfenning and Conal Elliot, *Higher-Order Abstract Syntax*, ACM SIGPLAN'88 Conference on Programming Language Design and Implementation, 199–208, 1988.

[9] D. Friedman and M. Wand, *Reification: Reflection without Metaphysics*, Proc. ACM Symposium on LISP and Functional Programming, 348–355, 1984.

[10] Gudmund S. Frandsen and Carl Sturtivant, *What is an Efficient Implementation of the λ-calculus?*, Proc. ACM Conference on Functional Programming and Computer Architecture (J. Hughes, ed.), LNCS 523, 289–312, 1991.

[11] J. Steensgaard-Madsen, *Typed representation of objects by functions*, TOPLAS 11-1, 67–89, 1989.

[12] N.G. de Bruijn, *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation*, Indagationes Mathematicae 34, 381–392, 1972.

[13] H.P. Barendregt, *Discriminating coded lambda terms*, in (A. Anderson and M. Zeleny eds.) Logic, Meaning and Computation, Kluwer, 275–285, 2001.

[14] François-Nicola Demers and Jacques Malenfant, *Reflection in logic, functional and object-oriented programming: a Short Comparative Study*, Proc. IJCAI Workshop on Reflection and Metalevel Architectures and their Applications in AI, 29–38, 1995.

[15] Daniel P. Friedman, Mitchell Wand, and Christopher T. Haynes, *Essentials of Programming Languages – 2nd ed*, MIT Press, 2001.

[16] Oleg Mazonka and Daniel B. Cristofani, *A Very Short Self-Interpreter*, `http://arxiv.org/html/cs.PL/0311032`, 2003.

[17] D. Hofstadter, *Godel, Escher, Bach: an Eternal Golden Braid*, Basic Books, Inc., 1979.

[18] H.P. Barendregt, *The Lambda Calculus, its Syntax and Semantics*, revised edition, North-Holland, Amsterdam, 1984.

[19] Simon Peyton Jones, *Tackling the awkward squad: monadic input/output,*

concurrency, exceptions, and foreign-language calls in Haskell, in "Engineering theories of software construction", ed. Tony Hoare, Manfred Broy, Ralf Steinbruggen, IOS Press, 47–96, 2001.

[20] The Haskell Home Page, `http://haskell.org/`.

[21] Brainfuck homepage, `http://www.muppetlabs.com/~breadbox/bf/`.

[22] Torben Æ. Mogensen, *Linear-Time Self-Interpretation of the Pure Lambda Calculus*, Higher-Order and Symbolic Computation 13(3), 217-237, 2000.

[23] D. A. Turner, *Another algorithm for bracket abstraction*, J. Symbol. Logic 44(2), 267–270, 1979.

[24] J. T. Tromp, `http://www.cwi.nl/~tromp/cl/Lambda.lhs`, 2004.

Philosophy

Chapter 15

Where Do New Ideas Come From? How Do They Emerge? Epistemology as Computation (Information Processing)

Gordana Dodig-Crnkovic

Mälardalen University, Västerås, Sweden,
gordana.dodig-crnkovic@mdh.se

This essay presents arguments for the claim that in *the best of all possible worlds* (Leibniz) there are sources of unpredictability and creativity for us humans, even given a pancomputational stance. A suggested answer to Chaitin's questions: "*Where do new mathematical and biological ideas come from? How do they emerge?*" is that they come from the world and emerge from basic physical (computational) laws. For humans as a tiny subset of the universe, a part of the new ideas comes as the result of the re-configuration and reshaping of already existing elements and another part comes from the outside as a consequence of openness and interactivity of the system. For the universe at large it is randomness that is the source of unpredictability on the fundamental level. In order to be able to completely predict the Universe-computer we would need the Universe-computer itself to compute its next state; as Chaitin already demonstrated there are incompressible truths which means truths that cannot be computed by any other computer but the universe itself.

15.1. Introduction

The previous century had logical positivism and all that emphasis on the philosophy of language, and completely shunned speculative metaphysics, but a number of us think that it is time to start again. There is an emerging digital philosophy and digital physics, a new metaphysics associated with names like Edward Fredkin and Stephen Wolfram and a handful of like-minded individuals, among whom I include myself.

It was in June 2005 I first met Greg Chaitin at the E-CAP 2005 conference in Sweden, where he delivered the Alan Turing Lecture, and presented his book *Meta Math!* It was a remarkable lecture and a remarkable book

that has left me wondering, reading and thinking since then.[1]

The overwhelming effect was a feeling of liberation: we were again allowed to think big, think *système du monde,* and the one Chaitin suggested was constructed as digital philosophy – something I as a computer scientist and physicist found extremely appealing. God is a computer programmer, Chaitin claims, and to understand the world amounts to be able to program it!

Under these premises the theory of information, specifically Chaitin's algorithmic theory of information becomes a very elegant and natural way to reconstruct epistemology, as demonstrated in Chaitin (2006). The epistemological model that is according to Chaitin central to algorithmic information theory is that a scientific or mathematical theory is a computer program for calculating the facts, and the smaller the program, the better the theory. In other words, understanding is compression of information![2]

In exploring epistemology as information theory, Chaitin addresses the question of the nature of mathematics as our most reliable knowledge, illustrated by Hilbert's program for its formalization and automatization. Based on algorithmic information theory Chaitin comes to this enlightening conclusion:

> In other words, the normal, Hilbertian view of math is that all of mathematical truth, an infinite number of truths, can be compressed into a finite number of axioms. But there are an infinity of mathematical truths that cannot be compressed at all, not one bit!

This is a very important result, which sheds a new light on epistemology. It sheds a new light on the meaning of Gödel's and Turing's negative responses to Hilbert's program. What is scientific truth today after all,[3] if not even mathematics is able to prove every true statement within its own domain? Chaitin offers a new and encouraging suggestion – mathematics may be not as monolithic and a priori as Hilbert believed.

But we have seen that the world of mathematical ideas has infinite complexity; it cannot be explained with any theory having a finite number of

[1] I had the privilege to discuss the Turing Lecture article with Chaitin, while editing the forthcoming book Dodig-Crnkovic G. and Stuart S., eds. (2007), *Computation, Information, Cognition – The Nexus and The Liminal*, Cambridge Scholars Publishing. The present paper is meant as a continuation of that dialog.

[2] For a detailed implementation of the idea of information compression, see Wolff (2006).

[3] Tasic, in his *Mathematics and the Roots of Postmodern Thought* gives an eloquent answer to this question in the context of human knowledge in general.

bits, which from a sufficiently abstract point of view seems much more like biology, the domain of the complex, than like physics, where simple equations reign supreme.

The consequence is that the ambition of having one grand unified theory of mathematics must be abandoned. The domain of mathematics is more like an archipelago consisting of islands of truths in an ocean of incomprehensible and uncompressible information. Chaitin, in an interview in September 2003 says:

> You see, you have all of mathematical truth, this ocean of mathematical truth. And this ocean has islands. An island here, algebraic truths. An island there, arithmetic truths. An island here, the calculus. And these are different fields of mathematics where all the ideas are interconnected in ways that mathematicians love; they fall into nice, interconnected patterns. But what I've discovered is all this sea around the islands.

So, it seems that apart from Leibniz bewildering question quoted by Chaitin (2006): "Why is there something rather than nothing? For nothing is simpler and easier than something." (Leibniz, Section 7 of *Principles of Nature and Grace*), there is the following, equally puzzling one:

Why is that something which exists made of parts rather than in one single piece?

For there are two significant aspects of the world which we observe: the world exists, and it appears to us as divisible, made of parts. The parts, however, are not totally unrelated universes in a perfectly empty vacuum.[4] On the contrary, physical objects constitute myriads of intricate complex structures on many different scales, and as we view them through various optics we find distinct characteristic complex structures.

Starting from the constatation that our understanding of the world is fragmented, it is easy to adopt a biological paradigm and see human knowledge as an eco-system with many sub-systems with different interacting parts that behave like organisms. Even though an organism is an autonomous individual it is not an isolated system but a part of a whole interconnected living network.

[4] Here the interesting question of the nature of a vacuum is worth mentioning. A vacuum in modern physics is anything but empty – it is simmering with continuous activity, with virtual particles popping up from it and disappearing into it. Chaitin's ocean of the unknown can be imagined as a vacuum full of the activity of virtual particles.

Contrary to the common model of a computing mechanism, in which the computer given a suitable procedure and an input, sequentially processes the data until the procedure ends (i.e. the program halts) or a model of a physical system which is assumed to be hermetically isolated with all possible conservation laws in effect, a model of a biological system must necessarily be *open. A biological system is critically reliant on its environment for survival. Separate parts of an ecological system communicate and are vitally dependent on each other.*

To sum up, extremely briefly, Chaitin's informational take on epistemology, the world is for a human effectively an infinite resource of truths, many of them incompressible and incomprehensible. Mathematics is not a monolithic, perfect, eternal crystal of the definite true essence of the world. It is rather, like other sciences, a fragmented and open structure, living and growing as a complex biological adaptive eco-system.

In the conclusion of *Epistemology as Information Theory: From Leibniz To* Ω, Chaitin leaves us with the following assignment:

> In fact, I believe that this is actually the central question in biology as well as in mathematics; it's the mystery of creation, of creativity: Where do new mathematical and biological ideas come from? How do they emerge?

Normally one equates a new biological idea with a new species, but in fact every time a child is born, that's actually a new idea incarnating; it's reinventing the notion of "human being," which changes constantly.

"I have no idea how to answer this extremely important question; I wish I could. Maybe you will be able to do it. Just try! You might have to keep it cooking on a back burner while concentrating on other things, but don't give up! All it takes is a new idea! Somebody has to come up with it. Why not you?" (Chaitin 2006)

That is where I want to start. After reading *Meta Math*! and a number of Chaitin's philosophical articles,[5] and after having written a thesis based on the philosophy of computationalism/informationalism (Dodig-Crnkovic, 2006) I dare to present my modest attempt to answer the big question above, as a part of a Socratic dialogue. My thinking is deeply rooted in pancomputationalism, characterized by Chaitin in the following way:

[5] A goldmine of articles may be found on Chaitin's web page. See especially `www.cs.auckland.ac.nz/~chaitin/g.pdf`, *Thinking About Gödel & Turing.*

> And how about the entire universe, can it be considered to be a computer? Yes, it certainly can, it is constantly computing its future state from its current state, it's constantly computing its own time-evolution! And as I believe Tom Toffoli pointed out, actual computers like your PC just hitch a ride on this universal computation! (Chaitin 2006)

If computation is seen as information processing, pancomputationalism turns to paninformationalism. Historically, within the field of computing and philosophy, two distinct branches have been established: informationalism, in which the focus is on information as the stuff of the universe; (Floridi 2002, 2003 and 2004) and computationalism, where the universe is seen as a computer. Chaitin (2006) mentions the cellular automata researchers and computer scientists Fredkin, Wolfram, Toffoli, and Margolus, and the physicists Wheeler, Zeilinger, 't Hooft, Smolin, Lloyd, Zizzi, Mäkelä, and Jacobson, as the most prominent computationalists. In Dodig-Crnkovic (2006) I put forward a dual-aspect info-computationalism, in which the universe is viewed as a structure (information) in a permanent process of change (computation). According to this view, information and computation constitute two aspects of reality, and like the particle and wave, or matter and energy, capture different facets of the same physical world. Computation may be either discrete or continuous[6] (digital or analogue). The present approach offers a generalization of traditional computationalism in the sense that "computation" is understood as the process governing the dynamics of the physical universe.

Digital philosophy is fundamentally neo-Pythagorean especially in its focusing on software aspects of the physical universe (either code or a process). Starting from the pancomputationalist version of digital philosophy, epistemology can be naturalized so that knowledge generation can be explained in pure computationalist terms (Dodig-Crnkovic, 2006). This will enable us to suggest a mechanism that produces meaningful behavior and knowledge in biological matter and that will also help us understand what we might need in order to be able to construct intelligent artifacts.

[6]The universe is a network of computing processes and its phenomena are info-computational. Both continuous as discrete, analogue as digital computing are parts of the computing universe. (Dodig-Crnkovic, 2006). For the discussion about the necessity of both computational modes on the quantum mechanical level see Lloyd (2006).

15.2. Epistemology Naturalized by Info-Computation

Naturalized epistemology is an idea that the subject matter of epistemology is not our concept of knowledge, but knowledge as a natural phenomenon (Feldman, Kornblith, Stich, Dennett). In what follows I will try to present knowledge generation as natural computation, i.e. information processing. One of the reasons to taking this approach is that info-computationalism provides a unifying framework which makes it possible for different research fields such as philosophy, computer science, neuroscience, cognitive science, biology, and a number of others to communicate within a common framework.

In this account naturalized epistemology is based on the computational understanding of cognition and agency. This entails evolutionary understanding of cognition (Lorenz 1977, Popper 1978, Toulmin 1972 and Campbell et al. 1989, Harms 2004, Dawkins 1976, Dennett 1991). Knowledge is a result of the structuring of input data (data → information → knowledge) (Stonier, 1997) by an interactive computational process going on in the nervous system during the adaptive interplay of an agent with the environment, which increases agents' ability to cope with the world and its dynamics. The mind is seen as a computational process on an informational structure that, both in its digital and analogue forms, occurs through changes in the structures of our brains and bodies as a consequence of interaction with the physical universe. This approach leads to a naturalized, evolutionary epistemology that understands cognition as a phenomenon of interactive information processing which can be ascribed even to the simplest living organisms (Maturana and Varela) and likewise to artificial life.

In order to be able to comprehend cognitive systems we can learn from the historical development of biological cognitive functions and structures from the simple ones upward. A very interesting account of developmental ascendancy, from bottom-up to top-down control, is given by Coffman 2006. Among others this article addresses the question of the origin of complexity in biological organisms, including the analysis of the relationship between the parts and the whole.

15.3. Natural Computation beyond the Turing Limit

As a direct consequence of the computationalist view that every natural process is computation in a computing universe, "computation" must be

generalized to mean *natural* computation. MacLennan 2004 defines "natural computation" as "computation occurring in nature or inspired by that in nature", which besides classical computation also includes quantum computing and molecular computation, and may be represented by either discrete or continuous models. Examples of computation occurring in nature encompass information processing in evolution by natural selection, in the brain, in the immune system, in the self-organized collective behavior of groups of animals such as ant colonies, and in particle swarms. Computation inspired by nature includes genetic algorithms, artificial neural nets, simulated immune systems, and so forth. There is a considerable synergy gain in relating human-designed computing with the computing in nature. Here we can illustrate Chaitin's claim that "we only understand something if we can program it": In the iterative course of modeling and computationally simulating (programming) natural processes, we learn to reproduce and predict more and more of the characteristic features of the natural systems.

Classical ideal theoretical computers are mathematical objects and are equivalent to algorithms, abstract automata (Turing machines or "logical machines" as Turing called them), effective procedures, recursive functions, or formal languages. Contrary to traditional Turing computation, in which the computer is an isolated box provided with a suitable algorithm and an input, left alone to compute until the algorithm terminated, interactive computation (Wegner 1988, Goldin et al. 2006) presupposes interaction i.e. communication of the computing process with the environment during computation. Interaction consequently provides a new conceptualization of computational phenomena which involves communication and information processing. Compared with new emerging computing paradigms, in particular with interactive computing and natural computing, Turing machines form the proper subset of the set of information processing devices. (Dodig-Crnkovic, 2006, paper B)

The Wegner-Goldin interactive computer is conceived as an open system in communication with the environment, the boundary of which is dynamic, as in living biological systems and thus particularly suitable to model natural computation. In a computationalist view, organisms may be seen as constituted by computational processes; they are "living computers". In the living cell an info-computational process takes place using DNA, in an open system exchanging information, matter and energy with the environment.

Burgin (2005) in his book explores computing beyond the Turing limit and identifies three distinct components of information processing systems: hardware (physical devices), software (programs that regulate its functioning and sometimes can be identical with hardware, as in biological computing), and infoware (information processed by the system). Infoware is a shell built around the software-hardware core, which is the traditional domain of automata and algorithm theory. Semantic Web is an example of infoware that is adding a semantic component to the information present on the web (Berners-Lee, Hendler and Lassila, 2001).

For the implementations of computationalism, interactive computing is the most appropriate general model of natural computing, as it suits the purpose of modeling a network of mutually communicating processes (Dodig-Crnkovic 2006). It will be of particular interest to computational accounts of epistemology, as a cognizing agent interacts with the environment in order to gain experience and knowledge. It also provides a unifying framework for the reconciliation of classical and connectionist views of cognition.

15.4. Cognitive Agents Processing Data → Information → Knowledge

Our specific interest is in how the structuring from data to information and knowledge develops on a phenomenological level in a cognitive agent (biological or artificial) in its interaction with the environment. The central role of interaction is expressed by Görzel (1994) in the following way:

> Today, more and more biologists are waking up to the sensitive environment-dependence of fitness, to the fact that the properties which make an organism fit may not even be present in the organism, but may be emergent between the organism and its environment.

One can say that living organisms are "about" the environment, that they have developed adaptive strategies to survive by internalizing environmental constraints. The interaction between an organism and its environment is realized through the exchange of physical signals that might be seen as data, or when structured, as information. Organizing and mutually relating different pieces of information results in knowledge. In that context, computationalism appears as the most suitable framework for naturalizing epistemology.

Maturana and Varela (1980) presented a very interesting idea that even the simplest organisms possess cognition and that their meaning-production apparatus is contained in their metabolism. Of course, there are also non-metabolic interactions with the environment, such as locomotion, that also generates meaning for an organism by changing its environment and providing new input data. We will take Maturana and Varelas' theory as the basis for a computationalist account of evolutionary epistemology.

At the physical level, living beings are open complex computational systems in a regime on the edge of chaos,[7] characterized by maximal informational content. Complexity is found between orderly systems with high information compressibility and low information content and random systems with low compressibility and high information content. Living systems are "open, coherent, space-time structures maintained far from thermodynamic equilibrium by a flow of energy". (Chaisson, 2002)

Langton has compared these different regions to the different states of matter. Fixed points are like crystals in that they are for the most part static and orderly. Chaotic dynamics are similar to gases, which can be described only statistically. Periodic behavior is similar to a non-crystal solid, and complexity is like a liquid that is close to both the solid and the gaseous states. In this way, we can once again view complexity and computation as existing on the edge of chaos and simplicity. (Flake 1998)

Artificial agents may be treated analogously with animals in terms of different degrees of complexity; they may range from software agents with no sensory inputs at all to cognitive robots with varying degrees of sophistication of sensors and varying bodily architecture.

The question is: how does information acquire meaning naturally in the process of an organism's interaction with its environment? A straightforward approach to naturalized epistemology attempts to answer this question via study of evolution and its impact on the cognitive, linguistic, and social structures of living beings, from the simplest ones to those at highest levels of organizational complexity (Bates 2005).

[7]Bertschinger N. and Natschläger T. (2004) claim "Employing a recently developed framework for analyzing real-time computations we show that only near the critical boundary such networks can perform complex computations on time series. Hence, this result strongly supports conjectures that dynamical systems which are capable of doing complex computational tasks should operate near the edge of chaos, i.e. the transition from ordered to chaotic dynamics."

Various animals are equipped with varying physical hardware, sets of sensory apparatuses goals and behaviors. For different animals, the "aboutness" concerning the same physical reality is different in terms of causes and their effects.

Indeed, cognitive ethologists find the only way to make sense of the cognitive equipment in animal is to treat it as an information processing system, including equipment for perception, as well as the storage and integration of information; that is, after all, the point of calling it cognitive equipment. That equipment which can play such a role confers selective advantage over animals lacking such equipment no longer requires any argument. (Kornblith 1999)

An agent receives inputs from the physical environment (data) and interprets these in terms of its own earlier experiences, comparing them with stored data in a feedback loop. Through that interaction between the environmental data and the inner structure of an agent, a dynamical state is obtained in which the agent has established a representation of the situation. The next step in the loop is to match the present state with goals and preferences (saved in an associative memory). This process results in the anticipation of what various actions from the given state might have for consequences (Goertzel 1994). Compare with Dennett's (1991) Multiple Drafts Model. Here is an alternative formulation:

> This approach is not a hybrid dynamic/symbolic one, but interplay between analogue and digital information spaces, in an attempt to model the representational behavior of a system. The focus on the explicitly referential covariation of information between system and environment is shifted towards the interactive modulation of implicit internal content and therefore, the resulting pragmatic adaptation of the system via its interaction with the environment. The basic components of the framework, its nodal points and their dynamic relations are analyzed, aiming at providing a functional framework for the complex realm of autonomous information systems (Arnellos et al. 2005)

Very close to the above ideas is the interactivist approach of Bickhard (2004), and Kulakov & Stojanov (2002). On the ontological level, it involves naturalism, which means that the physical world (matter) and mind are integrated, mind being an emergent property of a physical process, closely related to the process metaphysics of Whitehead (1978).

15.5. Evolutionary Development of Cognition

Evolutionary development is the best known explanatory model for life on earth. If we want to understand the functional characteristics of life, it is helpful to reveal its paths of development.

One cannot account for the functional architecture, reliability, and goals of a nervous system without understanding its adaptive history. Consequently, a successful science of knowledge must include standard techniques for modeling the interaction between evolution and learning. (Harms, 2005)

A central question is thus what the mechanism is of the evolutionary development of cognitive abilities in organisms. Critics of the evolutionary approach mention the impossibility of "blind chance" to produce such highly complex structures as intelligent living organisms. Proverbial monkeys typing Shakespeare are often used as an illustration. However, Lloyd 2006 mentions a following, first-rate counter argument, originally due to Chaitin and Bennet. The "typing monkeys" argument does not take into account the physical laws of the universe, which dramatically limit what can be typed. The universe is not a typewriter, but a computer, so a monkey types random input into a computer.

Quantum mechanics supplies the universe with "monkeys" in the form of random fluctuations, such as those that seeded the locations of galaxies. The computer into which they type is the universe itself. From a simple initial state, obeying simple physical laws, the universe has systematically processed and amplified the bits of information embodied in those quantum fluctuations. The result of this information processing is the diverse, information-packed universe we see around us: programmed by quanta, physics give rise first to chemistry and then to life; programmed by mutation and recombination, life gave rise to Shakespeare; programmed by experience and imagination, Shakespeare gave rise to Hamlet. You might say that the difference between a monkey at a typewriter and a monkey at a computer is all the difference in the world. (Lloyd 2006)

Allow me to add one comment on Lloyd's computationalist claim. The universe/ computer on which a monkey types is at the same time the hardware and the program, in a way similar to the Turing machine. An example from biological computing is the DNA where the hardware (the molecule) is at the same time the software (the program, the code). In general, each new input restructures the computational universe and changes the

preconditions for future inputs. Those processes are interactive and self-organizing. That makes the essential speed-up for the process of getting more and more complex structures.

15.6. Informational Complexity of Cognitive Structures

Dynamics lead to statics, statics leads to dynamics, and the simultaneous analysis of the two provides the beginning of an understanding of that mysterious process called mind. (Görtzel 1994)

In the info-computationalist vocabulary, "statics" (structure) corresponds to "information" and "dynamics" corresponds to "computation".

One question which may be asked is: why doesn't an organism exclusively react to data as it is received from the world/environment? Why is information used as building blocks, and why is knowledge constructed? In principle, one could imagine a reactive agent that responds directly to input data without building an informational structure out of raw input.

The reason may be found in the computational efficiency of the computation concerned. Storage of data that are constant or are often reused saves huge amounts of time. So, for instance, if instead of dealing with each individual pixel in a picture, we can make use of symbols or patterns that can be identified with similar memorized symbols or patterns, the picture can be handled much more quickly.

Studies of vision show that cognition focuses on that part of the scene which is variable and dynamic, and uses memorized data for the rest that is static (this is the notorious frame problem of AI). Based on the same mechanism, we use ideas already existing to recognize, classify, and characterize phenomena. Our cognition is thus an emergent phenomenon, resulting from both memorized (static) and observed (dynamic) streams. Forming chunks of structured data into building blocks, instead of performing time-consuming computations on those data sets in real time, is an enormously powerful acceleration mechanism. With each higher level of organization, the computing capacity of an organism's cognitive apparatus is further increased. The efficiency of meta-levels is becoming evident in computational implementations. Goertzel illustrates this multilevel control structure by means of the three-level "pyramidal" vision processing parallel computer developed by Levitan and his colleagues at the University of Massachusetts. The bottom level deals with sensory data and with low-level processing such

as segmentation into components. The intermediate level handles grouping, shape detection and such; and the top level processes this information "symbolically", constructing an overall interpretation of the scene. This three-level perceptual hierarchy appears to be an exceptionally effective approach to computer vision.

We look for those objects that we expect to see and we look for those shapes that we are used to seeing. If a level 5 process corresponds to an expected object, then it will tell its children [i. e. sub-processes] to look for the parts corresponding to that object, and its children will tell their children to look for the complex geometrical forms making up the parts to which they refer, et cetera. (Görtzel 1994)

Human intelligence is indivisible from its presence in a body (Dreyfus 1972, Gärdenfors 2000, 2005, Stuart 2003). When we observe, act and reason, we relate different ideas in a way that resembles the relation of our body with various external objects. Cognitive structures of living organisms are complex systems with evolutionary history (Gell-Mann 1995) evolved in the interaction between first proto-organisms with the environment, and evolving towards more and more complex structures which is in a complete agreement with the info-computational view, and the understanding of human cognition as a part of this overall picture.

15.7. Conclusions

This essay attempts to address the question posed by Chaitin (2006) about the origin of creativity and novelty in a computational universe. For that end, an info-computationalist framework was assumed within which information is the stuff of the universe while computation is its dynamics. Based on the understanding of natural phenomena as info-computational, the computer in general is conceived as an open interactive system, and the Classical Turing machine is understood as a subset of a general interactive/adaptive/self-organizing universal natural computer. In a computationalist view, organisms are constituted by computational processes, implementing computation *in vivo*.

All cognizing beings are physical (informational) systems in constant interaction with their environment. The essential feature of cognizing living organisms is their ability to manage complexity, and to handle complicated environmental conditions with a variety of responses that are results of adaptation, variation, selection, learning, and/or reasoning. Increasingly

complex living organisms arise as a consequence of evolution. They are able to register inputs (data) from the environment, to structure those into information, and, in more developed organisms, into knowledge. The evolutionary advantage of using structured, component-based approaches (data → information → knowledge) is improving response time and the computational efficiency of cognitive processes.

The main reason for choosing an info-computationalist view for naturalizing epistemology is that it presents a unifying framework which enables research fields of philosophy, computer science, neuroscience, cognitive science, biology, artificial intelligence and number of others to communicate, exchange their results and build a common knowledge. It also provides the natural solution to the old problem of the role of representation, a discussion about two seemingly incompatible views: a symbolic, explicit and static notion of representation versus implicit and dynamic (interactive, neural-network-type) one. Within info-computational framework, those classical (Turing-machine type) and connectionist views are reconciled and used to describe different levels or aspects of cognition.

So where do new mathematical and biological ideas come from? How do they emerge?

It seems to me that as a conclusion we can confidently say that they come from the world. Humans, just as other biological organisms, are just a tiny subset of the universe, and the universe has definitely an impact on us. A part of the new ideas is the consequence of the re-configuration and reshaping of already existing elements in the biosphere, like in component-based engineering. Life learns from both, from already existing elements and from something that comes from the outside of our horizon.
Even if the universe is a huge (quantum mechanical) computer for us it is an infinite reservoir of new discoveries and surprises. For even if the universe as a whole would be a totally deterministic mechanism, for humans to know its functioning and predict its behavior would take infinite time, as Chaitin already demonstrated that there are incompressible truths. In short, in order to be able to predict the Universe-computer we would need the Universe-computer itself to compute its next state.

That was my attempt to argue that in *the best of all possible worlds (*"*le meilleur des mondes possibles*" – Leibniz 1710) there are sources of creativity and unpredictability, for us humans, even given a pancomputational stance. I have done my homework.

15.8. Acknowledgements

I would like to thank Greg Chaitin for his inspiring ideas presented in his Turing Lecture on epistemology as information theory and the subsequent paper, and for his kindness in answering my numerous questions.

References

Arnellos, A., Spyrou, T. and Darzentas, J. "The Emergence of Interactive Meaning Processes in Autonomous Systems", In: Proceedings of FIS 2005: Third International Conference on the Foundations of Information Science. Paris, July 4-7, 2005.

Bates, M. J. "Information and Knowledge: An Evolutionary Framework for Information Science". Information Research 10, no. 4 (2005), `InformationR.net/ir/10-4/paper239.html`.

Bertschinger, N. and Natschläger, T. "Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks", Neural Comp. 16 (2004) 1413–1436.

Berners-Lee, T., Hendler, J. and Lassila, O. "The Semantic Web". Scientific American, 284, 5, (21001), 34–43.

Bickhard, M. H. "The Dynamic Emergence of Representation". In H. Clapin, P. Staines, P. Slezak (Eds.) Representation in Mind: New Approaches to Mental Representation, (2004), 71–90. Amsterdam: Elsevier.

Burgin, M. (2005) *Super-Recursive Algorithms*, Berlin: Springer.

Campbell, D. T. and Paller, B. T. "Extending Evolutionary Epistemology to "Justifying" Scientific Beliefs (A sociological rapprochement with a fallibilist perceptual foundationalism?)." In Issues in evolutionary epistemology, edited by K. Hahlweg and C. A. Hooker, (1989) 231–257. Albany: State University of New York Press.

Chaisson, E.J. (2001) *Cosmic Evolution. The Rise of Complexity in Nature.* Harvard University Press, Cambridge.

Chaitin, G. J. (1987) *Algorithmic Information Theory*, Cambridge University Press.

Chaitin, G "Epistemology as Information Theory", Collapse, (2006) Volume I, 27–51. Alan Turing Lecture given at E-CAP 2005, `www.cs.auckland.ac.nz/~chaitin/ecap.html`.

Chaitin, G. J. (1987) *Information Randomness & Incompleteness: Papers on Algorithmic Information Theory*, World Scientific.

Chaitin, G. J. (2003) Dijon Lecture, `www.cs.auckland.ac.nz/~chaitin/dijon.html`.

Chaitin, G. J. (2005). *Meta Math!: The Quest for Omega*. Pantheon.

Coffman, A. J. "Developmental Ascendency: From Bottom-up to Top-down Control", Biological Theory Spring 1, 2 (2006), 165–178.

Dawkins, R. 1976, 1982. *The Selfish Gene*. Oxford University Press.

Dennett, D. (1995), *Darwin's Dangerous Idea*, Simon & Schuster.

Dennett, D. (1991) *Consciousness Explained*. Penguin Books.

Dodig-Crnkovic, G. (2006) *Investigations into Information Semantics and Ethics of Computing*, Mälardalen University Press.

Dreyfus, H. L. (1972) *What Computers Can't Do: A Critique of Artificial Reason*. Harper & Row.

Flake, G. W. (1998) *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*, MIT Press.

Floridi, L. (2002) "What is the Philosophy of Information?", Metaphilosophy 33, 1-2, 123–145.

Floridi, L. (2003) *Blackwell Guide to the Philosophy of Computing and Information*, Oxford: Blackwell.

Floridi, L. (2004) "Open Problems in the Philosophy of Information", Metaphilosophy, 35, 4, 554–582.

Fredkin, E. (2003) "An Introduction to Digital Philosophy", International Journal of Theoretical Physics 42, 2, 189–247.

Gärdenfors, P. (2000) *Conceptual Spaces*, Bradford Books, MIT Press.

Gärdenfors, P., Zlatev, J. and Persson, T. "Bodily mimesis as 'the missing link' in human cognitive evolution", Lund University Cognitive Studies 121, Lund. 2005.

Gell-Mann, M. (1995) *The Quark and the Jaguar: Adventures in the Simple and the Complex*. Owl Books.

Görtzel, B. (1993) *The Evolving Mind*. Gordon and Breach.

Görtzel, B. (1994) *Chaotic Logic*. Plenum Press.

Goldin, D., Smolka S. and Wegner P. eds. (2006) *Interactive Computation: The New Paradigm*, to be published by Springer-Verlag

Harms, W. F. "Naturalizing Epistemology: Prospectus 2006", Biological Theory 1(1) (2006), 23–24.

Harms, W. F. *Information and Meaning in Evolutionary Processes*. Cambridge University Press, 2004.

Kornblith, H. (1999) "Knowledge in Humans and Other Animals". Noûs 33 (s13), 327.

Kornblith, H. ed. (1994) *Naturalizing Epistemology*, second edition, Cambridge: The MIT Press.

Kulakov, A. and Stojanov, G. "Structures, Inner Values, Hierarchies And Stages: Essentials For Developmental Robot Architecture", 2nd International Workshop on Epigenetic Robotics, Edinbourgh, 2002.

Leibniz, G. W. *Philosophical Papers and Letters*, ed. Leroy E. Loemaker (Dodrecht, Reidel, 1969).

Lloyd, S (2006) Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos, Alfred A. Knopf.

Lorenz, K. (1977) *Behind the Mirror.* London: Methuen.

MacLennan, B. "Natural computation and non-Turing models of computation", Theoretical Computer Science 317 (2004) 115 – 145.

Maturana, H. (1980) Autopoiesis and Cognition: The Realization of the Living. D. Reidel.

Maturana, H. and Varela, F. (1992) *The Tree of Knowledge.* Shambala.

Popper, K. R. (1972) *Objective Knowledge: An Evolutionary Approach.* Oxford: The Clarendon Press.

Stich, S. (1993) "Naturalizing Epistemology: Quine, Simon and the Prospects for Pragmatism" in C. Hookway & D. Peterson, eds., Philosophy and Cognitive Science, Royal Inst. of Philosophy, Supplement no. 34 (Cambridge University Press) p. 1-17.

Stonier, T. (1997) *Information and Meaning. An Evolutionary Perspective*, Berlin: Springer.

Stuart, S. (2003) "The Self as an Embedded Agent", Minds and Machines, 13 (2): 187.

Tasic, V. (2001) Mathematics and the Roots of Postmodern Thought. Oxford University Press.

Toulmin, S. (1972) *Human Understanding: The Collective Use and Evolution of Concepts.* Princeton University Press.

Wegner, P. "Interactive Foundations of Computing", Theoretical Computer Science 192 (1998) 315-51.

Whitehead, A. N. (1978) *Process and Reality: An Essay in Cosmology.* New York: The Free Press.

Wolff, J. G. (2006) Unifying Computing and Cognition, `Cognition Research.org.uk`, `www.cognitionresearch.org.uk/books/sp_book/ISBN0955072603_e3.pdf`.

Wolfram, S. (2002) *A New Kind of Science.* Wolfram Science.

Chapter 16

The Dilemma Destiny/Free–Will

F. Walter Meyerstein

Calle Tavern 45, 08006 Barcelona, Spain; fwm@filnet.es

The following brief considerations intend to relate the headline terms of this volume, namely *randomness* and *complexity,* with a dilemma that has occupied a central place in many philosophical systems throughout the ages. Admitting that the fate (destiny) of the world, at least in that part of it inhabited by conscious humans, is rigidly predetermined from its inception, what role, if any, is left for human freely decided actions?

And what is the alternative to a causally structured world, if not chaotic randomness making its development entirely unpredictable? As shown here, philosophers such as Leibniz have never found a valid escape-route to these rather unpalatable options. Surprisingly, however, Chaitin's work casts a brilliant new light on this conundrum!

Randomness and complexity are terms defining a situation in the world that has been quite difficult to admit by humans: the ideal of humanity seems to have always been to live in a predictable world, one in which the past could be comprehended, at least in part, and the future would not lie entirely beyond its capabilities. Of course, there was always the recourse to God: His infinite wisdom does oversee the complex development of the world, from its inception until its end. This rather easy escape did not satisfy most philosophers, who tried to find and establish *laws and rules*; mathematical if at all possible, that would allow a partial insight into the working of the universe. The universe, then, would not be random for God – of course – but neither might it be entirely random for humans.

One of the most famous of these rules was put forward by *G. W. Leibniz* at the end of the XVII century: roughly stated, it says that in the world nothing happens without a *sufficient reason*, that is, if some entity pos-

sessed the necessary knowledge it would be able to give a reason sufficient to determine why things are as they are and not different and, interestingly, would enable that entity to answer the question: 'why is there something rather than nothing?', why is the world complex – something – rather than simple – *i.e.*, nothing? Note that a complex world may still be a 'good' world, wherein everything is disposed for the best – see below-, but in a random world anything can happen, the good as well as the evil. Consequently, to evict randomness from the creation is fundamental.

Leibniz was a strict believer. Thus, for him, the world is God's creation. In the infinity of *causal* chains that constitute the world, entangled as these chains may be, Leibniz distinguishes those that follow necessary connections and those that are contingent. The former are such that their opposite implies a contradiction; the latter can exist or not exist. But, in Leibniz' system, according the principle of sufficient reason, contingent causes owe their existence to the principle of what is best, the best then being the sufficient reason of things. God can do everything that is possible, but He will do only what is best. The result: the world, as it is, is the best of all possible worlds! Voltaire, in *Candide*, made imperishable fun of Leibniz' *optimistic* effort to justify God's creation, to evict, if not complexity, but randomness from His creation.

For each of the infinity of causal chains that constitute the universe, two further principles apply. They can be considered to be consequences of the principle of sufficient reason, although their justification by Leibniz is a rather involved affair. They are the 'law of the identity of indiscernibles' and the 'law of continuity'. The law of the identity of indiscernibles states that either a thing is wholly meaningless, and in that case cannot be distinguished from any other, or it is the sum of all or some of the predicates that can be supposed to be applicable to this subject. Thus, two things which are materially diverse always differ as to their predicates. However, these predicates may be infinite in number, as in this system the present state of a thing has a relation to all past and future states. Whereas the analysis of necessary connections, such as in the number system, comes to an end, as the analysis of numbers ends with unity, this is not the case of contingents or existents, that can be or cannot be. Here the analysis, Leibniz assures, goes to infinity without ever reaching primitive elements: necessary and contingent truths differ as rational numbers and surds. In fact, as is the case with surd ratios, their reduction to commensurable numbers involves an infinite process, and yet approaches a common measure, obtaining a

definite but unending series. It follows that the world of contingents, the world of what can possibly exist or not exist, involves infinity. Note further that for Leibniz space and time have not real existence: indeed, on this principle he bases his assertion that space and time are distinguished only by means of the predicates of things and not the other way around. The final result is encapsulated in Leibniz' dictum that there are no two drops of water perfectly alike. The law of continuity, on the other hand, asserts that the causal chains of things form a series, so that every possible intermediate between the first and the last term is filled once, and only once.

Philosophers who have endeavored to establish the strict rules by which the world is governed always had to face a vexing problem: how to accommodate *human free-will* in their world-system, that is, how can humans, uniquely in the entire creation, interfere with the pre-established (by God?) causal chains and start, so to say *ex nihilo*, a brand-new sequence of world-events? And how did Leibniz reconcile his thorough-going causal determinism with human free-will? His solution does not appear to be very brilliant: he proposes that God has communicated to us a certain degree of his perfection and of his liberty. But how then, in this best of all possible worlds, in this 'pre-established harmony', do some obviously evil actions of free humans fit? Note that admitting a free-will, if it is really *free*, is equivalent to introducing randomness into the world, as freedom of decision implies previous indetermination of the choices. In fact, the unsolved problem of conciliating a deterministic world that imposes a pre-established *destiny* on humans with a free-will is a very old one as I will now briefly show.

In around 44 BC, probably after the assassination of Caesar, and shortly before he himself was murdered, *M. T. Cicero* wrote an essay with the title *De Fato* (On Fate) from which only a part has come down to us. The subject of this essay constitutes the analysis of the relation of human free-will with a rigid destiny as resulting from universal causation à la Leibniz. To show how this embarrassing contradiction was understood in the first century BC, I here cite a few passages from this work, using the translation of H. Rackham in the Loeb Classical Library.

The essay is written as a dialogue between different parties, who in turn quote previous philosophers to bolster their respective argumentations. Citing Carneades, head of the Platonic Academy in the second century BC, we read: "If everything takes place with antecedent causes, all events take place in a closely knit web of natural interconnections; if this is true, all things are caused by necessity; if this is true, nothing is in our power. But

something is in our power. Yet if all events take place by fate, there are antecedent causes of all events. Therefore it is not the case that whatever events take place take place by fate (XIV 31)". But we read further on: "Even if it is admitted that nothing can happen without an antecedent cause, what good would that be unless it be maintained that the cause in question is a link in an eternal chain of causation? But a cause is that which makes the thing of which it is the cause come about". However, absurd situations result: "For on these lines a well-dressed traveller also will be said to have been the cause of the highwayman's robbing him of his clothes (XIV 34)".

From the extant fragments it rather clearly transpires that the problem of destiny and free-will also escaped Cicero. A later philosopher, A.. Gellius (second century AD), wrote this comment in his *Noctes Atticae*: "In the book that he wrote on the subject of fate Marcus Cicero says that th[is question] is very obscure and involved, and he remarks that the philosopher Chrysippu (Stoic philosopher of the third century BC), finding himself quite at sea in the difficulty of how to explain his combination of universal fatalism with human free-will, ties himself up in a knot."

These very succinct brushstrokes may give you an idea of how, throughout the centuries, philosophers have struggled to unravel the mystery posed by the randomness and complexity of the world, how they have endeavoured to reintroduce order, purpose, even a *design* into it, and how they have tried to make mutually compatible the visibly contradictory concepts of fate (destiny) and human free-will. Amazingly, but in fact not so surprising, the fascinating work of *G. S. Chaitin*, as exposed in many of his books, but particularly in his 2005 *META MATH!*, directly impinge on these questions, as I will now try to show.

First of all, let me remark that *causality* is an extremely difficult idea as is corroborated by the many books and papers on this subject by recent philosophers. However, here I assume causality to be a clear term, intuitively understood. Further, note that the causal chains determining destiny are supposed to be continuous, each individual link connected uniquely to a past and a future event: prior links are assumed to be the sufficient cause of all posterior links. Apparently, only in this way these philosophers have understood a rigidly pre-determined fate (*cf.* Leibniz' law of continuity). It is obvious that any different 'solution' introduces randomness into the world making a rigid destiny an impossibility. Also note that cross-linking or other interactions of the causal chains are not taken into consideration.

If it is assumed that a cause uniquely engenders its consequences or its effects, it might also be admitted that such a cause, now taken as the input to a suitable computer, results in that computer outputting all the effects of said cause. And thus, right away, we are in Chaitin's territory! Instead of a causal chain we can now speak of a Formal Axiomatic System (FAS), the axioms of which describe the cause(s) at hand and the theorems of which correspond to its possible consequences. An immediate corollary of this consideration is: the starting link(s), the shortest expression of the primary cause(s) that generates *all* these causal chains is of irreducible complexity; not unexpected, as it is equivalent to God.

The idea of the philosophers postulating a fate or destiny is basically encapsulated in Leibniz' idea of a sufficient reason for everything that happens. In other words, as back in time (and, theoretically, as beyond in the future) as is possible for humans to consider, the causal destiny chain works as a FAS. However, in *META MATH!* Chaitin writes: "... formal axiomatic systems are a failure! Theorems proving algorithms do not work". Because what makes no sense is "mindlessly and mechanically grinding away deducing all the possible consequences of a fixed set of rules and ideas" (page146). These remarks by Chaitin are made in a totally different context from the subject here treated: destiny and free-will. But they strongly support the view that the idea of 'mechanically deducing' the effects of an antecedent cause leads nowhere. Why?

The main reason is this: these immensely long causal chains correspond to equally long binary sequences when one admits their being equivalent to the output of a suitable computer. Consequently, with probability approaching 1, *i.e.*, certainty, they are irreducible random sequences, not compressible into any system of axioms substantially shorter than themselves. To make *all* causal chains determining the destiny of some human being to show such an order, to make them such that they can be algorithmically compressed into a short axiom system, is out of the question. Imagine now the same dream as applied to the causal chains affecting, for all times, *all* humanity! And note that not even God can disentangle this fabulous mess. This is one of the most surprising results that springs from the analysis of Chaitin's marvellous number Omega. The binary sequence constituting Omega, resulting from a neat mathematical definition, are such that, assuming you could determine the first N bits of that sequence, you will never be in a position to predict whether the bit $N+1$ is a 0 or a 1 with a

probability better than1/2. And this corresponds to the probability of the outcome of a toss of a fair coin, the paradigm of randomness: the bits of Omega constitute a maximally random sequence.

But to what corresponds free-will in this approach? A human's entirely free *decision* only exists between two equally probable alternatives and thus can only be accomplished by tossing a fair coin! (Unless he or she chooses to starve like Buridan's ass!). Of course, if the probabilities of two choices facing some individual are unequal, the freedom of the will of this individual is compromised and cannot anymore said to be entirely free. On the other hand, assuming that humans make entirely free decisions which in every case start a brand-new causal chain, we run again into trouble, as the set of all these free coin tosses constitute – by definition as it were – an irreducible random collection. In other words, it would not be possible to express the free decisions of humans by means of a compact, simple axiom system. In sum, the old dilemma of the assumption of human free-will in a world subject to a rigid causal destiny cannot be resolved: the world is just too complex, randomness cannot be evicted, this may be a habitable world, but probably it is not the best.

Chapter 17

Aliquid Est Sine Ratione: On some Philosophical Consequences of Chaitin's Quest for Ω

Ugo Pagallo
Law School, University of Turin
Turin, Italy; ugo.pagallo@unito.it

Introduction

In order to examine some philosophical consequences of Gregory Chaitin's quest for Ω this paper comes in five sections. First of all (first section), I consider Chaitin's interpretation of Leibniz's thought and how Chaitin's halting probability Ω invalidates Leibniz's principle of sufficient reason. Then (second section), I compare this analysis with a classic reading of Leibniz, namely what Heidegger states in *The Principle of Reason*. Once we have grasped Heidegger's criticism to the principle *nihil est sine ratione* (section 3), I will stress some paramount differences between Heidegger's thesis and Chaitin's theorems (section 4). By showing some flaws in the German scholar's viewpoint, what I would like to lay emphasis on is the impact of Chaitin's results in contemporary philosophical debate (section 5).

17.1. Leibniz's legacy

It is not so hard to understand why Chaitin has paid so much attention to Leibniz's philosophy in the last years. The German philosopher can be considered the father—at least, the grandfather—of contemporary digital philosophy. Leibniz invented binary arithmetic and in the early 1670s, before he had to go back to Germany as counsellor of princes, built some of the first calculating machines that he personally displayed in Paris and at

the Royal Society in London. As a kind of Pythagorean,[1] Leibniz conceived the world mathematically as a peculiar "progression" of 0s and 1s in order to represent the world according to the simplest hypothesis: *Omnibus ex nihil ducendis sufficit unum.*[2] This position is much stronger than, say, Occam's razor as far as Leibniz's key idea somehow anticipates Chaitin's perspective on complexity, that is the outlook "in which at the same time the hypotheses are as simple as possible, and the phenomena are as rich as possible."[3]

However, this sort of simplification is fruitful when we bring facts back to "truths of reason:" that is, when we deal with the dominion of logic and mathematics—Leibniz would add metaphysics—so that truth can be reduced to a finite set of principles or axioms considered "eternal truths."[4]

But what about daily life where we—the "informavores"—are designed by evolution to be "epistemically hungry seekers of information, in an endless quest to improve our purchase on the world"? [5] In other words, what about "contingent truths" as Leibniz would have said?

According to Chaitin, "Leibniz's solution to the problem is to claim that contingent events are also true for a reason, but in such cases there is in fact an infinite series of reasons, an infinite claim of cause and effect, that while utterly beyond the power of human comprehension is not at all beyond the power of comprehension of the divine mind."[6] Hence, from a metaphysical standpoint, Leibniz claimed that contingent propositions, although not necessary, should also be thought of *a priori*: In fact, everything that happens in the world must have its reason. While it is easy to imagine what stroke "the great Arnauld" when the French scholar read Leibniz's letters on free will and human destiny,[7] it is the principle of reason that would enable us to consider even contingent events as comprehensible, logical, and determinable. Leibniz's suggests that God created the best of all possible worlds and, in doing so, He selected an infinite series of "hy-

[1] Remember Leibniz's maestro Erhard Weigel in Jena and the portrait by Pietro della Vecchia now at the Chrysler Museum in Washington. Further details in U. Pagallo [2005, 39–40].
[2] Leibniz's phrase in G. Chaitin [2005, 61].
[3] Again, Leibniz's quotation (and translation) in G. Chaitin [2005, 63].
[4] This idea is sponsored by Leibniz since his first legal works: See for example § 83 of *De Arte combinatoria* (1666).
[5] I am quoting D.C. Dennett [2003, 93] whose thesis are discussed infra § 4.
[6] G. Chaitin [2005, 120].
[7] I comment these letters in U. Pagallo [2005, 111–122].

pothetical necessities" which let us free to choose between, say, a 0 or a 1. As in Leibniz's favourite example of Julius Caesar crossing the Rubicon river, this is a sound fact—as the Roman Senate fully understood—so that it also represents a (historical) truth and, *therefore*, should be thought of as demonstrable. (By the way, this idea is shared by many contemporary scholars. In a nutshell, if the universe appears to be "fort composée," it is still determined by evolution,[8] but it is so complex that leaves Man with his "free will."[9])

Nonetheless, how can we decide if we have rendered sufficient reasons for something? How can we measure the complexity of the universe and, in particular, its contingent truths?

When Leibniz discusses in sections V and VI of the *Discours de métaphysique* how to tell apart a world in which science works from one where it does not, he considers the case of a mathematical equation for any given finite set of points on a paper. It allows you to define when you have a proper scientific law, that is when the law itself appears "simple." Yet, it would still not be possible to distinguish random from rule bound points because there is always a formula in order to "pass through" those very points. So, what Chaitin's algorithmic theory of information aims to do is precisely to refine such ideas by adding two new elements: "First, we measure complexity in terms of bits of information, i.e., 0s and 1s. Second, instead of mathematical equations, we use binary computer programs. Crucially, this enables us to compare the complexity of a scientific theory (the computer program) with the complexity of the data it explains (the output of the computer program)."[10]

The well-known result is Chaitin's halting probability Ω: By linking the complexity of the law to the data it attempts to explain, it is possible to prove facts—namely the independent mathematical facts of Ω—that are "true for no reason."[11] Indeed, from a simple mathematical definition you get what is both logically and computably irreducible, maximally unknowable. Randomness is what Leibniz was wrong about in, at least, the realm of mathematical truths, for we cannot think of all of them as demonstrable. (Here, Chaitin deepens and updates an aspect of Leibniz's thought

[8] Cf. for example S. Wolfram [2002], whose thesis Chaitin have frequently discussed in recent work.
[9] See again D.C. Dennett [2003].
[10] G. Chaitin [2004, 2].
[11] Cf. G. Chaitin [2004, 3] and [2004, 133].

that Gödel's theorems of incompleteness should have made easier to understand.)

But, again, what about contingent truths? Leibniz's problem on deciding when sufficient reasons have been given appears quite similar to Chaitin's question about how to decide if a computer program is "elegant." In order to shed some further light on this connection, let us proceed with a strange German misunderstanding: Heidegger's, not Hilbert's!

17.2. A German misunderstanding

Martin Heidegger, one of the most influential philosophers of the 20^{th} Century, dedicated his 1955–1956 course at the University of Freiburg to Leibniz's principle *nihil est sine ratione*.[12] These lectures are of great importance not only because they offer a canonical reading of Leibniz, but also because they focus on the Principle of Sufficient Reason, i.e., as Leibniz himself explained to Arnauld in a letter on July 14^{th} 1686: "nothing happens without a reason that one can always render as to why the matter has run its course this way rather than that."[13]

Heidegger portrays Leibniz in a twofold way. On the one hand, Leibniz is presented as a milestone in the development of modern logic into logistics: "Only through looking back on what Leibniz thought can we characterise the present age—an age one calls the atomic age—as an age pervasively bepowered by the power of the *principium rationis sufficientis*."[14] On the other hand, Leibniz should be understood at the light of German idealism and of its metaphysical credo on the infinite self-knowing of the absolute spirit.[15] While Heidegger links the principle of reason to something he calls "the Destiny of Being",[16] it would not be a mere coincidence if Leibniz, a German thinker, expressly posited that "nothing is without reason" only 23 centuries after incessant Western philosophical tradition. Indeed, metaphysics would be complete and philosophy accomplished at the very moment in which nothing escapes from the quest for reason as in the case of German idealism (from Leibniz to Hegel, so to say). "What is mighty about the principle of reason displays its power in that the *principium reddendae rationis*—to all appearances only a Principle of cognition—also counts, pre-

[12] I will refer to Reginald Lilly's edition as M. Heidegger [1991].
[13] The letter quoted in M. Heidegger [1991, 119].
[14] M. Heidegger [1991, 33].
[15] See for example M. Heidegger [1991, 65].
[16] *Geschick* in German. We will give further details in section 3.

cisely in being the fundamental principle of cognition, as the Principle for everything that is."[17]

Yet, there is a fundamental paradox in Leibniz's principle: either the principle according to which everything-has-a-reason has itself a reason, or we are dealing with the only principle which is not affected by what the principle says. The way out of the problem could be assume it as an axiom for *axiomata sunt propositiones, quae ab omnibus pro manifestis habentur*, i.e., axioms are principles that are held by everyone to be obvious. We can in fact say that nothing is without a reason since it is equivalent to affirm that nothing happens without a cause or, from a logical standpoint, any demonstration that tries to avoid the principle of sufficient reason would fall in contradiction. All in all, the paradox is also present in Chaitin's quest for Ω because you will always need reasons in any critique of the principle of reason!

However, according to Heidegger, the axiomatic method suggested by Leibniz does not tell us anything about the meaning of Reason and, worst of all, it leads directly to the realm of mere prudential calculus. "It would be both short-sighted and presumptuous if we wanted to disparage modern axiomatic thinking. But it would also be a childish and pathetic notion if we were to believe that this modern thinking would let itself be bent back upon its great and open origin in the thinking of the Greeks."[18] What Heidegger claims is that modern science responds to the demand of rendering reasons without the capacity of mediating such a demand. Reflective thinking shrinks into instrumental reasoning because the principle of providing sufficient reasons reduces beings to simple objects of calculation. Since it would not be possible to think of the world and Man outside of the complete technicizing of the atomic era, the key word that sums up the entire process is *information*. I.e., "information at one and the same time means the appraisal that as quickly, comprehensively, and profitably as possible acquaints contemporary humanity with the securing of its necessities, its requirements, and their satisfaction."[19] To be our first computer scientist—as Leibniz might be understood today—would mean to forget the principle of being that grounds the principle of reason, that is, in Heidegger's jargon, the direction from whence the principle of reason "speaks" in both axiomatic and informational way.

[17] M. Heidegger [1991, 23].
[18] M. Heidegger [1991, 20].
[19] M. Heidegger [1991,124].

Of course, there are lots of misunderstandings in Heidegger's remarks: Ad instance, he speaks as if, fifty years ago, neither the limits of formal axiomatic systems nor the polysemy of the very concept of information were clear.[20]

But what I want to stress is something else. When Heidegger states that "the unique unleashing of the demand to render reasons threatens everything of humans' being-at-home and robs them of the roots of their substance,"[21] he is really trying to show the limits of axiomatic thinking and calculability, limits that can be seen only from the perspective of "Being." From this viewpoint, what Heidegger discovers seems to be very similar to what Chaitin proves, namely the Principle of Insufficient reason. But the way Heidegger gets there is quite different. Before comparing their conclusions, let us look at the reasons Heidegger gives in order to show the limits of the very principle of reason.

17.3. The darkest of modern thoughts

According to Heidegger, we can think outside or beyond the realm of Leibniz's principle thanks to a "leap." This would come from an understanding of the Principle "nothing is without reason" in which the connection of the words 'is' and 'reason' is more important than the link between 'nothing' and 'without.' Heidegger distinguishes in fact between beings and Being: The first—beings – is assimilated with the connection between 'nothing' and 'without.' The second category—Being—is considered to represent the other couple: 'is'/'reason'. Heidegger holds as "obvious" or "necessary" that "all beings have a ground/reason,"[22] so the leap occurs in the case of Being. According to the German philosopher, this way we get the ground/reason itself, i.e., the reason of the foundation that grounds but has no ground as "Being reigns from out of the essence of ground/reason. (...) Therefore being can never first have a ground/reason which would supposedly ground it. (...) To the extent that being as such grounds, it remains groundless. 'Being' does not fall within the orbit of the principle of reason, rather only beings do."[23]

[20] For example, Claude Shannon's tripartite approach claims it would be necessary to distinguish the technical problems regarding the quantification of communication from the semantic issues concerning meaning and truth, let alone the impact and effectiveness of information on human behaviour.
[21] M. Heidegger [1991, 31].
[22] Cf. M. Heidegger [1991, 110 and 125].
[23] M. Heidegger [1991, 51].

Leibniz's principle of reason therefore would not fully pass the test that the other basic principle we use in any demonstration passes, that is Aristotle's principle of contradiction which is presented in the fourth book of *Metaphysics* as a "not hypothetical principle" (Γ 3, 1005 b 14). Again, the motive depends on the leap that occurs from the principle of reason as foundation of beings (nothing/without) to being qua being "that is, qua ground/reason."[24] We do not search for the ground/reason of the very principle of reason and, hence, we avoid the *regressio ad infinitum*, only if we admit a principle that gives us measures but still remains immeasurable. We are dealing with a principle that grounds without having itself a ground and that let you calculate and provide reasons while remaining itself incalculable. The more you try to provide reasons for everything—including Being— the less you understand that the very principle of reason is nothing but "an uttering of being."[25] By reversing the perspective, Heidegger thus speaks about the groundless "abyss" of what is necessarily incalculable, immeasurable, groundless, otherwise it would not be possible to get any grounds, reasons, measures, or computations of scientific reasoning. To be explicit: "Being, as what grounds, has no ground; as the abyss it plays the play that, as *Geschick*, passes being and ground/reason to us."[26]

On this very basis, the aim of the German philosopher is to represent our era as the outcome and triumph of Western philosophical tradition. The first attempt to quantify relations between words, signs, and things—namely Leibniz's research on the *characteristica universalis*—has led to a world in which individuality vanishes at breakneck speed into total uniformity and all depends on the provision of atomic energy to establish Man's domination over the World. The overall idea is that "for the first time a realm is opened up which is expressly oriented toward the possibility of rendering the ground of beings. (. . .) This epoch characterises the innermost essence of the age we call modernity."[27] Heidegger's critique of modernity and of its "granddaughter," i.e., contemporary technology in the atomic era, is thus grounded on the limits of "the Mighty Principle" for it was this very principle (is/reason) that turned out to guarantee calculability of objects (nothing/without) for instrumental cognition. The diagnosis is hence complete: While the perfection of technology is only a simple echo of the

[24] M. Heidegger [1991, 53].
[25] l M. Heidegger [1991, 49].
[26] M. Heidegger [1991, 113].
[27] M. Heidegger [1991, 55].

required completeness of rendering reasons, this very completeness (nothing/without) ends up in the oblivion of *aliquid*, namely Being, which *est sine ratione* (ground/reason). In Heidegger's phrase: "only beings have—and indeed necessarily—a ground/reason. A being is a being only when grounded. However, being, since it is itself ground/reason, remains without ground/reason."[28] That is why 'something' is but has no reason: *aliquid sine ratione*. Q.E.D.

17.4. A Post-modern turning point?

Heidegger has become the well-known hero of the contemporary philosophical approach which insists on the limits of instrumental reasoning and the end of metaphysics. Somehow this does not sound all too foreign from Chaitin's own speculation. For example, in the conclusions of his *Meta Math!* we read that formal axiomatic systems are a failure for coping with creativity since "no mechanical process (rules of the game) can be really creative, because in a sense anything that ever comes out was already contained in your starting point."[29] Furthermore, with the case of the flip of a coin Chaitin highlights what rationalism violently opposes. Indeed, for a rationalist everything happens for a reason in the physical world and in the world of mathematics everything is true for a reason. But, an infinite series of independent tosses of a fair coin represents "a horrible nightmare for any attempt to formulate a rational world view (...) because each outcome is a fact that is true for no reason, that's true only by accident!"[30] (By the way, before exploring any connection between the halting probability Ω and, say, quantum mechanics,[31] this very idea was exploited in order to prove that determinism would be perfectly compatible with the principle that some things have no reason at all. In a nutshell, the idea of flipping a coin "accomplishes just the opposite of digitising in computers: Instead of absorbing all the micro-variation in the universe, it amplifies it, guaranteeing that the unimaginably large sum of forces acting at the moment will tip the digitiser into one of two states, heads or tails, but with no salient necessary conditions for either state.")[32]

However, there is a paramount difference between Heidegger's idea of

[28] M. Heidegger [1991, 125].
[29] G. Chaitin [2005, 142].
[30] G. Chaitin [2005, 119].
[31] See ad instance C. Calude [2002], and with M.A. Stay [2005].
[32] D.C. Dennett [2003, 85].

something without a reason and Chaitin's theorems. While the German philosopher posits "something" as Principle, Chaitin presents it as facts related to 'beings:' This, of course, offers a much stronger position than, say, Occam's Razor or Leibniz's idea of simplicity, for this perspective avoids the slippery field of 'is/reason' by showing the flaws of 'nothing/without.' In particular, by proving that there are things without reason—namely the independent mathematical facts of Ω—Chaitin shows why Heidegger was wrong in conceding that all beings have necessarily a ground/reason. There is no need to accept this in order to preserve the exception of the groundless Principle! Indeed, 'is/reason' may vary: You could grasp it in a Dao way,[33] or define it more precisely as axioms that "are exactly equal in size to the body of interesting theorems."[34] At any rate, it is as if the foggy realm of metaphysical principles would have contaminated the beings in the world. What still remains metaphorical in Heidegger's jargon on 'is/reason' as what is deprived of ground, becomes scientific with Chaitin's definition of randomness. He has in fact picked out from "the infinitely dark blackness of random reals" a single one which is properly 'something/without' reason.[35]

Needless to say that *this* version of the principle of Insufficient reason—what mathematicians call incompleteness—prevent us from some exaggerations of most post-modern philosophers who, in the wake of Heidegger's thesis, pretend "to leap-into-the-abyss." In spite of the German philosopher, Chaitin's post-modern thesis, namely his "quasi-empirical" approach in Maths shows how axiomatic thinking "let[s] itself be bent back upon its origin in the cosmos of the Greeks."[36] In fact, contrarily to most post-modern philosophers, Chaitin uses sound reasoning, namely meta-mathematics, in order to demonstrate the limits of reason when it is forced to approach what is both logically and computably irreducible, maximally unknowable. This version of the principle of Insufficient reason, however, suggests a further question: As far as Chaitin's proofs concern the world of pure Maths, how are they connected to the real world? Does God really play dice?

Here we have to distinguish between two fields: One has been adequately studied by Chaitin and concerns physics and even biology. The other is given by social sciences and contemporary debate on, for example,

[33] For this peculiar connection see G. Parkes [1987]. Further theoretical details in U. Pagallo [2005b].
[34] G. Chaitin [2005, 118].
[35] Cf. G. Chaitin [2005, 140].
[36] Consider again M. Heidegger [1999, 20].

determinism and free will.[37] In both cases, however, you have the problem of pointing out, so to speak, 'something/without' reason in the real world: is that possible?

17.5. Hyper-modernity and some concluding remarks

Let me introduce you to the conclusions of this paper by summing up the results we have already got: Leibniz's principle of reason is untenable because Chaitin shows something that is true for no reason simpler than itself. Contrarily to Heidegger and to part of the contemporary philosophical thought the mistake is not then in 'is/reason,' namely scientific reasoning, but in 'nothing/without' related to beings. That would automatically mean for Leibniz that some contingent events are groundless or without reason: for Chaitin, "this mean[s] that physical randomness, coin tossing, something non-mechanical, is the only possible source of creativity."[38] This version of the principle of Insufficient reason as 'something/without' has emerged in contemporary debate on true randomness in the sense of indeterminism. On the one hand, Chaitin recalls Karl Svozil's opinion "that some new, deeper, hidden-variable theory will eventually restore determinacy and law to physics."[39] On the other, some scholars like Daniel Dennett would add that you do not really need indeterminism, that is real randomness, in order to vouchsafe free will and get 'something/without' cause.[40]

Chaitin has explained his viewpoint on what is 'something/without' reason in physics and why he interprets unpredictability and chaos—that is: real intrinsic randomness—of quantum mechanics in a digital way.[41] However, by adopting this perspective of 0s and 1s as a measure of complexity and information we shed further light on a significant issue in the traditional philosophical debate as well as in social sciences. If determinism is to be compatible with 'something/without' cause, you do not have to grasp the complexity of social institutions as infinite—as occurs in mathematics—in order to get the same results of incompleteness.[42] In particular, it is pretty clear that truth can be represented as a sub-set of environmental complexity also in social sciences. This is the case of legal systems with textbook

[37] See for example the essays edited by R. Kane [2002].
[38] G. Chaitin [2005, 142].
[39] G. Chaitin [2005, 120].
[40] D.C. Dennett [2003, 90].
[41] Cf. G. Chaitin [2005, 91–93].
[42] Full details in U. Pagallo [2006b].

examples of *fons extra ordinem* like revolutions or "illegal customs" in European civil law or like the Bill of Indemnity-tradition in UK's common law. As suggested by Hayek's ideas and contemporary research on social evolution, the principle of Insufficient reason states that 'something/without' cause does exist in human interaction for, otherwise, there would be no third fruitful way between mere chaos and old historical determinism such as Marx's social laws or Hegel's philosophy of history.[43]

However, to admit that some things have no reason, is by no way to admit a sort of narrative post-modern reasoning. Again, the mistake is not in 'is/reason' as a demand of ground: The point is how to construe the connection between 'ground/reason' and 'something/without' in a digital way. If Heidegger presented Leibniz as responsible of the evils of modernity, Chaitin thinks of Leibniz as the main precursor of Today's digital philosophy-paradigm and, indeed, it took more than two centuries to grasp all of his hints! Once we have stressed the difference between Heidegger's version of the principle of insufficient reason—that is Being without ground—and Chaitin's version of 'something/without,' it is hereby clear why one should follow the second path. Instead of traditional metaphysics or pure speculations, we get truths that are such for no reason simpler than themselves. So, to be aware of the limits which the principle of reason encounters and, hence, to understand the strength of the principle of Insufficient reason through computable irreducibility and the maximally unknowable—both in Maths and social sciences—does not lead to post-modernism but to what I would like to call "Hyper-modernity." Indeed, Heidegger was right when he claimed that it is reason as scientific reason, namely modernity which unveils the core of contemporary E-revolution and technology. However, the German thinker failed to comprehend that it is only through scientific reason that we focus on the very limits of reason itself. After Chaitin's analysis on both digital ontology and digital epistemology, it is then time to apply his theorems to metaphysics. The quest for Ω is correlated to the traditional inquiry of 'being qua being:' *Aliquid est sine ratione.*[44]

[43] I analyse Chaitin's contribution to contemporary debate in adaptive systems (as in Murray Gell-Mann's work), or in social complex systems (as in Mark C. Taylor's research), in G. Chaitin [2006, 87–92].

[44] It is noteworthy that there is no such contribution in an introduction like Loux-Zimmerman [2003]. As far as I know the first work on "digital metaphysics" is by E. Steinhart, *Digital metaphysics*, in *The Digital Phoenix: How Computers are Changing Philosophy*, edited by T. Bynum and J. Moor, Basil Blackwell: New York 1998, pp. 117–134.

References

C. Calude [2002]: *Information and Randomness*, Springer: Berlin.

C. Calude, M.A. Stay [2005]: From Heisenberg to Gödel via Chaitin. In *International Journal of Theoretical Physics*, 44, 7, pp. 1053–1065.

G. Chaitin [2004]: *Leibniz, Information, Math and Physics*. In: *Wissen und Glauben/Knowledge and Belief. Akten des 26. Internationalen Wittgenstein-Symposiums 2003*, Winfried/Weingartner: Wien, pp. 277–286.

G. Chaitin [2005]: *Meta Math! The Quest for Omega*, Pantheon: New York.

G. Chaitin [2006]: *Teoria algoritmica della complessità*, Giappichelli: Torino.

D.C. Dennett [2003]: *Freedom Evolves*, Penguin: London.

M. Heidegger [1991]: *The Principle of Reason*, edited by Reginald Lilly, Indiana University Press: Bloomington and Indianapolis.

R. Kane [2002]: *The Oxford Handbook of Free Will*, edited by Robert Kane, Oxford University Press: New York.

M.J. Loux, D. W. Zimmerman [2003]: *The Oxford Handbook of Metaphysics*, edited by Michael J. Loux and Dean W. Zimmerman, Oxford University Press: New York.

U. Pagallo [2005a]: *Introduzione alla filosofia digitale. Da Leibniz a Chaitin*, Giappichelli: Torino.

U. Pagallo [2005b]: Plato's Daoism and the Tübingen School. In: *Journal of Chinese Philosophy*, 32, 4, pp. 597–613.

U. Pagallo [2006a]: *Teoria giuridica della complessità*, Giappichelli: Torino.

U. Pagallo [2006b]: Chaitin e le scienze pratiche della complessità. In: Chaitin [2006, 79–102].

G. Parkes [1987]: *Heidegger and Asian Thought*, edited by Graham Parkes, University of Hawaii Press: Honolulu.

S. Wolfram [2002]: *A New Kind of Science*, Wolfram Media: Champaign Ill.

Essays

Chapter 18

Proving and Programming

Cristian S. Calude[α 1], Elena Calude[β] and Solomon Marcus[γ]

[α] *Department of Computer Science, University of Auckland, New Zealand;* cristian@cs.auckland.ac.nz

[β] *Institute of Mathematics and Information Sciences, Massey University at Albany, New Zealand;* e.calude@massey.ac.nz

[γ] *Romanian Academy, Mathematics, Bucharest, Romania;* Solomon.Marcus@imar.ro

There is a strong analogy between proving theorems in mathematics and writing programs in computer science. This paper is devoted to an analysis, from the perspective of this analogy, of *proof* in mathematics. We will argue that while the Hilbertian notion of proof has few chances to change, future proofs will be of various types, will play different roles, and their truth will be checked differently. Programming gives mathematics a new form of understanding. The computer is the driving force behind these changes.

18.1. Introduction

The current paper, a continuation of [12], is devoted to an analysis of *proof* in mathematics from the perspective of the analogy between proving theorems in mathematics and writing programs in computer science. We will argue that:

(1) Theorems (in mathematics) correspond to algorithms and not programs (in computer science); algorithms are subject to mathematical proofs (for example for correctness).
(2) The role of proof in mathematical modelling is very small: adequacy is the main issue.
(3) Programs (in computer science) correspond to mathematical models. They are not subject to proofs, but to an adequacy and relevance anal-

[1] Corresponding author

ysis; in this type of analysis, some proofs may appear. Correctness proofs in computer science (if any) are not cost-effective.
(4) Rigour in programming is superior to rigour in mathematical proofs.
(5) Programming gives mathematics a new form of understanding.
(6) Although the Hilbertian notion of proof has few chances to change, future proofs will be of various types and will play different roles, and their truth will be checked differently.

18.2. Proving vs. programming: today

Aristotle introduced the concept of proof as an epistemological tool, to establish *absolutely certain knowledge*. The argument follows a sequence of rigourously defined steps, starting from "first principles", which are claimed to be *self-evident truths*, using rules which are *truth-preserving*. The original intention was to derive *certain conclusions*, but this goal seems to be too ambitious.

In mathematics, the first principles are called axioms, and the rules are referred to as deduction/inference rules. A proof is a series of steps based on the (adopted) axioms and deduction rules which reaches a desired conclusion. Every step in a proof can be checked for correctness by examining it to ensure that it is logically sound.

According to Hilbert:

> The rules should be so clear, that if somebody gives you what they claim is a proof, there is a mechanical procedure that will check whether the proof is correct or not, whether it obeys the rules or not.

While ideally sound, this type of proof (called *Hilbertian* or *monolithic* [21]) cannot be found in mathematical articles or books (except for a few simple examples). However, most mathematicians believe that almost all "real" proofs, published in articles and books, can, with tedious work, be transformed into Hilbertian proofs. Why? Because real proofs look *convincing* for the mathematical community [21]. Going further, DeMillo, Lipton and Perlis argued that real proofs should be highly non-monolithic because they aim to be heard, read, assimilated, discussed, used and generalised by mathematicians—they are part of a social process.

Deductive rules are truth-preserving, but although the conclusion, generically termed as theorem, yields *knowledge*[2], there is no claim that it yields *certain knowledge*. The reason is simple: nothing certifies the

[2]We may called this *deductive knowledge*.

truth of the axioms. The epistemic status of axioms is an interesting and troubling question. Relativity appears in many instances, from the plurality of geometries (Euclidean and non-Euclidean), to Gödel's incompleteness theorem and various independence results (Continuum Hypothesis). Mathematically, there is no most preferred geometry neither the true set theory. *If there are no self-evident truths, then there is no certain knowledge.* For Thurston [54]

> ... the foundations of mathematics are much shakier than the mathematics that we do.

Programming is the activity of solving problems with computers. It includes the following steps: a) developing the *algorithm* to solve the problem, b) writing the algorithm in a specific programming language (that is, coding the algorithm into a program), c) assembling or compiling the program to turn it into machine language, d) testing and debugging the program, e) preparing the necessary documentation.

Ideally, at d) one should have written: *proving the correctness of the algorithm*, testing and debugging the program. We said, "ideally", because correctness, although desired, is only practised in very few instances (for example, the programs involved in the proof of the Four-Color Theorem were not proved correct.)[3] In programming practice (in a commercial/industrial environment), correctness is seldom required and much less proved. Some reasons include: a) the fact that, in general, correctness is undecidable [11], so failure to prove correctness has little meaning, b) for most non-trivial cases, correctness is a monumental task which gives an added confidence at a disproportionate cost. There are many projects and products dedicated to automated testing of program correctness, for example, VIPER or the more recent TestEra (automated testing of Java programs). The current approach in software and hardware design involves a combination of empirical and formal methods aiming to get better and better programs.

18.3. Mathematical examples

We will analyse three mathematical problems which reflect the current evolution of mathematical proofs.

[3]See the proof [10, 53].

18.3.1. *The twin prime conjecture*

The twin prime conjecture states that there are infinitely many pairs of twin primes (i.e. pairs $(p, p+2)$, where p and $p+2$ are primes). To date, all attempts to solve the problem have failed. (Recently, Arenstorf's proof [2] had a serious error and the paper was retracted.) In 2005, Goldston, Pintz and Yildirim [29] proved that there are infinitely many primes for which the gap to the next prime is as small as possible when compared with the average gap between consecutive primes. This spectacular result comes very close to the twin prime conjecture, which asserts that, apart from the case of 2 and 3, the gap is the smallest possible, i.e. 2. The Goldston–Pintz–Yildirimir proof is explained in a recent paper by Soundararajan [52]. Here are the main questions discussed: Can we say anything about the statistical distribution of gaps between consecutive primes? Does every even number[4] occur infinitely many times as a gap between consecutive primes? How frequently do twin primes occur? Interesting for our discussion is the following comment [52] (p. 2):

> Number theorists believe they know the answers to all these questions but cannot always prove that the answers are correct.

Earlier, in 2003, Goldston and Yildirim announced that there are infinitely many primes such that the gap to the next prime is very small. The proof looked convincing till A. Granville and K. Soundararajan discovered a tiny flaw which looked fatal (see [23] for the story). The flaw was discovered not by carefully *checking* Goldston and Yildirim's proof, but by *extending* it to show that there are infinitely many primes such that the gap to the next prime is less than 12 (the gap-12 theorem), a result which was too close to the twin prime conjecture to be true: they didn't believe it! B. Conrey, the director of the American Institute of Mathematics, which was close to this work, is quoted by Devlin [23] by saying that, without the "unbelievable" Granville and Soundararajan gap-12 theorem,

> the Goldston-Yildirim proof would in all probability have been published and the mistake likely not found for some years.

How many proofs are wrong? Although many (most?) proofs are probably "incomplete or benignly wrong"—that is, they can be in principle fixed—it is almost impossible to make an educated guess about how many proofs are wrong. One reason is that many proofs are only superficially checked, because either they have limited interest or they never come to be used (or both).

[4]Every gap between primes is even except for 3 and 2.

18.3.2. *The Kepler conjecture*

In 1997, Thomas Hales [31] published a detailed plan—a mixture of mathematical proofs and extensive computer calculations (see also [55])—describing a new strategy for attacking the Kepler conjecture. Hales' full proof appears in a series of papers taking up more than 250 pages and gigabytes of computer programs and data [15], all posted on the Internet. The proof cannot be checked without running his programs. In the end, his paper [32] was published in Annals of Mathematics, in spite of the inconclusive verdict given by the panel of 12 referees (see more in [14]).

The semi-failure of the correctness-checking process motivated Hales to start the project Flyspeck [27] whose purpose is *to produce a formal proof of the Kepler Conjecture*. Hales estimates that the project will run for 20 years before reaching a final conclusion.

We are motivated to join T. Anderson [9] (p. 2389) in asking the question: *must we consign mathematics to the dustbin until computers have confirmed the validity of the theorems and proofs?*

18.3.3. *The Poincaré conjecture*

The Poincaré conjecture was stated in 1904 by Henri Poincaré [43]. Colloquially, it states that the three-dimensional sphere is the only type of bounded three-dimensional space possible that contains no holes. The first solution carries a $1 million prize, as one of the Millennium Problems of the Clay Mathematical Foundation [20].[5]

The most significant scientific achievement of 2006, the *Breakthrough of the Year*, was, according to *Science* magazine, the solution of the Poincaré conjecture by Grigory Perelman [35]:

> This year's Breakthrough salutes the work of a lone, publicity-shy Russian mathematician named Grigori Perelman, who was at the Steklov Institute of Mathematics of the Russian Academy of Sciences until 2005. The work is very technical but has received unusual public attention because Perelman appears to have proven the Poincaré Conjecture, a problem in topology whose solution will earn a $1 million prize from the Clay Mathematics Institute. That's only if Perelman survives what's left of a 2-year gauntlet of critical attack required by the Clay prize rules.[6]

[5] If you think that this is a lot of money, then refer to the BBC announcement—broadcast as we are writing this paper—confirming that "David Beckham will leave Real Madrid and join Major League Soccer side LA Galaxy at the end of the season"; he will be paid $1 million *per week* [4].

[6] Most mathematicians think he will.

Perelman's solution appears in a series of papers he circulated and posted on the Internet (not published yet?) in 2003 [40–42]. In 2006, his solution was officially recognised: Perelman was offered the Fields Medal. As is well-known (and publicised), Perelman declined the Fields Medal. Rumour has it that he has expressed little interest in the Clay prize too.

Beyond obvious journalistic aspects, the story is significant in at least two directions: a) it reinforces the social aspect of the process of doing mathematics, as the Clay prize rules stipulate that an acceptable solution has to resist the critical analysis of the mathematical community for no less than two years, and b) the community accepted a result which was only posted on the Internet. While a) is not surprising in the least, we may ask *whether b) is the starting of a new pattern in mathematical communication.*

18.4. Computer science examples

We continue with three examples from computer science.

18.4.1. *Intel's bug*

According to *Wikipedia*,

> A software bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result).

"Flaw Reported in New Intel Chip" made the headlines of the Technology/Cybertimes section of the *New York Times* on May 6, 1997:

> The Intel Corp. will not formally introduce its Pentium II microprocessor until Wednesday, but the World Wide Web is already buzzing with reports of a bug that causes the new chip to make errors in some complex mathematical calculations.

Only three years earlier, on December 20, 1994, Intel recalled its popular Pentium processor due to an "FDIV" bug discovered by Thomas Nicely, who was working on, guess what? He was calculating Brun's sum [38], the series formed with the reciprocal of twin primes:

$$\left(\frac{1}{3}+\frac{1}{5}\right)+\left(\frac{1}{5}+\frac{1}{7}\right)+\left(\frac{1}{11}+\frac{1}{13}\right)+\left(\frac{1}{17}+\frac{1}{19}\right)+\cdots<\infty.$$

Nicely worked with five PC's using Intel's 80486 microprocessor and a Pentium [37]. Comparing the results obtained with the old machines and

the new Pentium, he observed a discrepancy in the calculation of the reciprocals of the twin primes 824,633,702,441 and 824,633,702,443. Running various tests, he identified the source of error in the floating point hardware unit of the Pentium CPU. Twenty three other errors were found by Andreas Kaiser, while Tim Coe arrived at the simplest error instance: the division 4,195,835/3,145,727—which evaluates to 1.33382044 $\cdots$—appears on the Pentium to be 1.33373906$\cdots$ Coe's ultra-simple example moved the whole story from the Internet to *New York Times*.

In contrast with errors found in mathematical proofs, which remain within the realm of mathematical experts, computer bugs attract the attention of a larger audience. For example, on January 17, 1995, Intel announced that it will spend $475 million to cover the recall of its Pentium chip to fix the problem discussed above, a problem that may affect only a few users.

Can bugs be avoided? More to the point of this article, *can the use of rigorous mathematical proofs guarantee that software and hardware perform as expected?*

18.4.2. *From algorithms to programs*

Bloch [6] identifies a bug in the Java implementation of a standard binary search[7]. Here is Bloch's code:

```
1:     public static int binarySearch(int[] a, int key) {
2:         int low = 0;
3:         int high = a.length - 1;
4:
5:         while (low <= high) {
6:             int mid = (low + high) / 2;
7:             int midVal = a[mid];
8:
9:             if (midVal < key)
10:                 low = mid + 1;
11:             else if (midVal > key)
12:                 high = mid - 1;
13:             else
14:                 return mid; // key found
15:         }
```

[7]Apparently was only recently reported to Sun, persisting for nine years.

```
16:          return -(low + 1);  // key not found.
17:      }
```

The bug is identified in line 6

```
 6:              int mid =(low + high) / 2;
```

with the explanation that the average value is truncated down to the nearest integer, a statement which is *true* for integers, but *false* for "bounded integers". If the sum of `low + high` is higher than $2^{31} - 1$, then the value overflows to a negative value and stays negative by division to 2. How frequently can this situation appear? For arrays longer in length than 2^{30}—not uncommon for Google applications—the bug appears. Bloch [6] offers some fixes and an implicit complaint: *how come that the bug persisted so long when he, as a PhD student, was taught a correctness proof [5] of the binary search algorithm?* Finally, he asks the crucial question: "and now we *know* the binary search is bug-free, right?"

18.4.3. *Bugs everywhere and Hoare's question*

Computer bugs are, literally, everywhere and they may affect many users. Most important software companies maintain bug databases: `bugs.sun.com/bugdatabase/index.jsp`, `bugs.kde.org`, `MySQLBugs`, `bugzilla.mozilla.org`, `bugs.apache.org`, etc. Here is a model of how to report bugs at Sun:

> If we don't know about your problem, we can't fix it. If you've isolated a problem that you think we're causing, and you can't find it here, submit a bug! Make sure you include all the relevant information including all the details needed to reproduce the problem. Submissions will be verified and prioritized. (Please note that *bug fixes are not guaranteed.*)[8]

Bugs can be of different types, hence producing varying levels of inconvenience to the user of the program. The costs of some bugs may be almost incalculable. A bug in the code controlling the Therac-25 radiation therapy machine led to at least six deaths between 1985 and 1987 [60]. The European Space Agency's $1 billion prototype Ariane 5's first test flight on June 4, 1996, failed, with the rocket self-destructing 37 seconds after launch [3]; the reason was a software bug, arguably one of the most expensive bugs in history. More recently, a security flaw in PayPal was exploited by fraudsters to steal credit card numbers and other personal information

[8] Our Italics.

belonging to PayPal users (June 16, 2006); and the Y2K7 bug (January 3, 2007) affected Microsoft's preview version of Expression Design. Finally, a bug discovered by M. Schwartz [50] found no interest in the community, so he had to write a small script showing how to use it to get all the email addresses from members subscribing to a Google group; Google fixed the problem on January 5, 2007. Improperly coded software seems to have caused the Mars Global Surveyor failure in November 2006; in January 2007, NASA launched an investigation [62].

The list can easily be continued. *Wired* magazine maintains a history of the worst software bugs [28].

In spite of all the examples discussed above, bugs and faulty software have killed remarkably few people. They caused more embarrassment, nuisance, inconvenience, but many fewer catastrophes. Early in January 2007, a 6.7 earthquake in Taiwan produced serious interruptions in the internet in Asia [58]; this showed that the internet is far from shockproof, but consequences were, again, not catastrophic. Finally, one more example: Boeing 777, one of the most automatic fly-by-wire air-planes, has flown since 1995 without any crashes or serious problems. So, we can ask with Hoare [34] the question: *how did software get so reliable without proof?*

18.5. Proving vs. programming: tomorrow

18.5.1. *Theorems and programs*

The practice of programming, by and large, produces "discoursive knowledge", a knowledge resulting from computing. "Deductive knowledge", complementary to discoursive knowledge, can be obtained by the mathematical analysis of the program (in some given context). These notions of knowledge correspond to Dijkstra's approaches (see [24]) to programs: *postulational* and *operational*. Under the postulational approach, the program text is considered a mathematical object. The semantic equivalence of two programs means that they meet the same specification. According to the operational approach, reasoning about programs means building a computational model with respect to which the program text is interpreted as executable code.

According to Dijkstra:

> The tragedy of today's world of programming is that, to the extent that it reasons about programs at all, it does so almost exclusively operationally.

The argument? "With growing size or sophistication of the program, the operational argument quickly becomes impossible to carry through, and the general adherence to operational reasoning has to be considered one of the main causes of the persistence of the software crisis". Dijkstra believes that, ultimately, real programmers don't reason about their programs, "they rather get their substitute for intellectual satisfaction from not quite understanding what they are doing in their daring irresponsibility and from the subsequent excitement of choosing the bugs they should not have introduced in the first place".

The last quotation reminds us what Bertrand Russell said about mathematics and what Martin Heidegger wrote about science: "Wissenschaft denkt nicht" (Science does not think). For Dijkstra, the analoguous situation in mathematics is the distinction between formal and informal; perhaps, he had in mind situations such as the distinction between axiomatic and naive set theory.

It makes sense to prove the correctness of an algorithm, but *not* the correctness of a program as various authors have argued [21, 26]. Programs are analogues of mathematical models; they may be more or less adequate to code algorithms. Adequacy is a property which depends on many factors, from pure formal/coding ones to physical and engineering ones. One can even argue that a "correctness proof" for a program, if one could imagine such a thing, adds very little to one's confidence in the program. In Knuth's [36] words:

> Beware of bugs in the above code: I have only proved it correct, not tried it.

The computer science analogy of the operational–postulational distinction corresponds to the difference—considered already at the beginning of the 19th century—between mathematics understood as calculation and mathematics as qualitative conceptual reasoning. In the analogy between proving and programming, *theorems correspond to algorithms not programs; programs correspond to mathematical models.*

18.5.2. *Mathematics = proof?*

The role of proof in mathematical modelling is very small: *adequacy is the main issue!* As mathematical modelling is closer to coding algorithms into programs, selecting algorithms to code, designing specifications to implement, one can re-phrase the arguments against the idea of proof of correctness of programs [21, 26] as arguments against the idea of proof of correctness of mathematical models. Models evolve and become more and

more adequate to the reality they model: however, they are never *true*. Here is an illuminating description by Schwartz [49]:

> ...it may come as a shock to the mathematician to learn that the Schrödinger equation for the hydrogen atom ... is not a literally correct description of this atom, but only an approximation to a somewhat more correct equation taking account of spin, magnetic dipole, and relativistic effects; that this corrected equation is itself only an ill-understood approximation to an infinite set of quantum field-theoretical equations; and finally that the quantum filed theory besides diverging, neglects a myriad of strange-particle interactions whose strength and form are largely unknown. ... The physicist, looking at the original Schrödinger equation, learns to sense it ... and this sense inspires ...

For example, engineers use theorems by "plugging in" values and relying on some (physical) interpretations of the conclusion. This is what makes planes fly and bridges stand.

The modelling component of mathematics appears not only in applications, but also in the way mathematics develops new concepts. Many important notions in mathematics reached an accepted definition only after a long process of modelling, from an intuitive, pre-mathematical notion to a more precisely defined, formal one. In the end, the accepted definition is adopted as a thesis claiming its adequacy [51]. For example, "Weierstrass's thesis" is the statement that the intuitive notion of continuity is extensionally equivalent to the notions yielded by the now standard definitions of "continuous function". Other examples include: the "function thesis" (identification of a function with a set of ordered pairs), "Tarski's thesis" (identification of Tarski's definition of truth in a formalised language with the intuitive notion of truth), "the Church-Turing thesis", etc. None of these theses can be *proved*, but various analyses can conclude their degrees of plausibility/adequacy/applicability. Mathematics in both its practice and development is an "open-texture" [51].

18.5.3. *Checking proofs*

There are many new types of proofs: probabilistic, experimental or hybrid proofs [7, 8] (computation plus theoretical arguments). Reflecting somehow Jean-François Lyotard's "No truth without money", Zeilberger [56] has argued in favour of the transition from rigorous proofs to an "age of *semi-rigorous* mathematics, in which identities (and perhaps other kinds of theorems) will carry price tags" measured in the computer and human resources necessary to prove them with a certain degree of confidence.

Zeilberger [57] sees the evolution of mathematics as follows:

> The real work of us mathematicians, from now until, roughly, fifty years from now, when computers won't need us anymore, is to make the transition from human-centric math to machine-centric math as smooth and efficient as possible.

Let's do the following mental experiment: apply literally to mathematical practice Hilbert's requirement for proof stated in Section 18.2 (in logical terms, *the proofs of a theory form a computable set*). Then Anderson's question, posed at the end of Subsection 18.3.2, is not only not surprising, but should be answered in an affirmative way. This could be a reasonable motivation for the project Flyspeck.

Probabilistically checkable proofs are mathematical arguments that can be checked probabilistically by reading just a few of their bits. In the early 1990's it was proved that every proof can be effectively transformed into a probabilistically checkable proof with only a modest increase in the original proof length. However, the transformation itself was complex. Recently, a very simple procedure was discovered by Dinur; see the presentation [48].

Now, feeling a loss of certitude, we should remember that Thales was the first to stimulate his disciples to criticise his assertions. This tradition was later lost, but recovered with Galilei. With Thales and Galilei we learned that human knowledge is essentially conjectural (see also [45]). Should mathematics and computer science accept being guided by this slogan, or is it adequate only for the natural and social sciences?

18.5.4. *Communication and understanding*

Of course, no theorem is validated before it is communicated to the mathematical community (orally and, eventually, in writing). Manin states it clearly:

> Proof is not just an argument convincing an imaginary opponent. Not at all. Proof is the way we communicate mathematical truth.

However, as Rota [46] pointed out:

> One must guard, however, against confusing the *presentation* of mathematics with the *content* of mathematics.

Proofs have to be written on paper, which means proofs are *physical*. From this perspective, proofs depend upon the physical universe (see more

in [13]). We already have to cope with existing, extremely long proofs. What about proofs that are too long to be written down? They may exist in principle, but they are impossible to read.[9]

In order to be able to amplify human intelligence and prove more complicated theorems than we can now imagine, we may be forced to accept incomprehensible or only partially comprehensible proofs. We may be forced to accept the help of machines for mental, as well as physical, tasks.

If mathematics depends on physics and mathematics is the main tool to understand physics, don't we have some kind of circularity? One explanation is that, in Lakatos's words, "mathematics is quasi-empirical", as has been extensively discussed by Chaitin (see [16, 17]).

Does physical interference combined with the use of computers destroy the understanding of mathematical facts? One could know that a theorem is true, but not really understand it! For Chaitin [18] (p. xiii) this is not the case:

> To me, you understand something only if you can program it. (You, not someone else!)

Why? Because [19] (p. 30):

> ... programming something forces you to understand it better, it forces you to really understand it, since you are explaining it **to a machine**.

18.5.5. *Rigour: operational vs. conceptual*

Standards of rigour have changed throughout the history of mathematics, and not necessarily from less rigour to more rigour. For Bertrand Russell, certainty has to match in power religious faith: "I wanted certainty in the kind of way in which people want religious faith".

Serre was quoted [47] saying that mathematics is the only producer of "totally reliable and verifiable" truths. This opinion seems to be contradicted by both Knuth [36]:

> ... programming demands a significantly higher standard of accuracy. Things don't simply have to make sense to another human being, they must make sense to a computer.

and Thurston [54]:

[9] An exponentially long quantum proof cannot be written down, since that would require an exponential amount of "classical" paper, but a quantum mind could directly perceive the proof, [13].

> The standard of correctness and completeness necessary to get a program to work at all is a couple of orders of magnitude higher than the mathematical community's standard of valid proofs.
>
> When one considers how hard it is to write a computer program even approaching the intellectual scope of a good mathematical paper, and how much greater time and effort have to be put into it to make it "almost" formally correct, it is preposterous to claim that mathematics as we practice it is anywhere near formally correct.

But we can go further into the past. Old Greek mathematics, with Pythagoras, Plato and Euclid, was essentially conceptual and this is the reason why they were able to invent what we call today a mathematical proof. Babylonian mathematics was exclusively operational. The move from an operational to a conceptual attitude in computer programming is similar to the evolution from Babylonian to Greek mathematics.

Coming to more recent periods in the history of mathematics, we observe the strong operational aspect of calculus in the 18th century, in contrast with the move to the predominantly conceptual aspect of mathematical analysis in the 19th century. Euler is a king of operational mathematics; Riemann and Weierstrass express *per excellence* the conceptual attitude. The transition from the former to the latter is represented by giants such as Abel and Cauchy. When Cauchy believed that he had proved the continuity of the limit of a convergent sequence of continuous functions, Abel, with no ironical intention, wrote: "Il semble que le théorème de Monsieur Cauchy admet des exceptions."[10] But, at that moment, neither of them was able to invent the notion of uniform convergence and, as a matter of fact, neither convergence nor continuity was effectively clarified. Only the second half of the 19th century brought their full understanding, together with the idea of uniformity, either with respect to convergence or with respect to continuity. We see here all characteristic features of a transition period, the transition from the operational to the conceptual attitude.

To stress the two facets of Cauchy's mathematics, one belonging to the intuitive-operational, the other to the rigourous-conceptual attitude, let us recall that, despite the fact that Cauchy is undoubtedly the founder of the exact differential calculus in the modern sense, he is also the mathematician who was convinced that the continuity of a function implies its differentiability and hence that any continuous function can be geometrically represented. We had to wait for Weierstrass and Riemann to understand

[10] It seems that the theorem of Mr Cauchy admits exceptions.

the gap existing between some mathematical concepts and their intuitive representation.

However, this evolution does not concern only calculus and analysis. It can be observed in all fields of mathematics, although the periods in which the transition took place may be different for various fields. For instance, in algebraic geometry it took place only in the 20th century, with the work of Oscar Zarisky. In fact, any conceptual period reaches its maturity under the form of an operational one, which, in its turn, is looking for a new level of conceptual attitude. The whole treatise of Bourbaki is a conceptual reaction to an operational approach. Dirichlet's slogan asking to replace calculations with ideas should be supplemented with another, complementary slogan, requiring us to detect an algorithmic level of concepts.

Can we expect a similar alternation of attitudes in respect to programming? Perhaps it is too early to answer, taking into account that the whole field is still too young. The question is not only academic, as the project Flyspeck reminds us.

18.5.6. *Is it meaningful to speak about the truth of axioms?*

In Section 18.2 we argued that mathematical proofs do not produce certain knowledge; they produce *rational belief*. The epistemological value of a proof reside in the degree of belief of its axioms. What is then the value of proof? Is it meaningful to speak about the truth of axioms?

First, a few more words should be said about axioms and primitive terms. Euclid avoids reference to primitive terms, but they exist in his *Elements*, hidden by pseudo-definitions such as "We call point what has no parts". Only modern axiomatic systems make explicit reference to primitive terms. Obviously, programs too could not be conceived in the absence of some primitive terms. A similar remark is in order about axioms. To what extent is it meaningful to raise the question about the truth of some axioms? Semantics is a matter of interpretation of a formal system, which, in its turn, has some primitive terms and some axioms among its bricks. Circularity is obvious. Gödel's (true) statements that cannot be proved could not be conceived in the absence of the respective formal system, which in its turn has among its bricks some primitive terms and some axioms. Maybe we can refer to another way to understand meaning, a way avoiding Hilbert's itinerary? For instance, the way it is understood in C. S. Peirce's semiotics or in modern linguistics. But do they have the rigour

we expect from a mathematical approach?

The whole idea of a formal proof is strictly dependent of the idea of a formal system. Is it meaningful to raise the question whether the Zermelo–Fraenkel axioms for set theory are or not true? We adopt a convention and it is proved by Gödel that if these axioms are consistent, then they remain consistent by adding the choice axiom or the continuum hypothesis.

It is interesting to observe that some authors have supplemented the Zermelo–Fraenkel axioms with the foundation axiom (aiming to interdict Russell's sets which are elements of themselves), while more recently Aczel and Barwise [1] have replaced the foundation axiom with the anti-foundation axiom, where hypersets are allowed. An object A is a hyperset if there exists an infinite sequence $A(n)$ such that $A(1) = A$ and for each n, $A(n+1)$ is an element belonging to $A(n)$. In the particular case when $A(n) = A$ for each n, we get the Russell sets. So, an axiom was replaced by its negation and the resulting theory proved to be very interesting. It has applications in mathematics (non-standard analysis), computer science (data bases, logical modelling of non-terminating computational processes), linguistics and natural language semantics (situation theory), and philosophy (the Liar Paradox). The well-known scenario of non-Euclidean geometries proves to be once more valid.

18.6. Acknowledgements

We thank Douglas Bridges, Andreea Calude, Greg Chaitin, and Nick Hay for many discussions and suggestions that improved our paper.

References

[1] P.Aczel, J. Barwise. Non-well-founded sets, *Journal of Symbolic Logic* 54, 3 (1989), 1111–1112.

[2] R. F. Arenstorf. There Are Infinitely Many Prime Twins, `http://arxiv.org/abs/math.NT/0405509` 26 May 2004.

[3] Ariane Flight 501, `http://spaceflightnow.com/cluster2/000714feature/ariane501_qt.html`.

[4] Beckham agrees to LA Galaxy move, `http://news.bbc.co.uk/sport2/hi/football/6248835.stm`, January 11, 2007.

[5] J. Bentley. *Programming Pearls*, Addison-Wesley, New York, 1986; 2nd ed. 2000 (Chapter 5).

[6] J. Bloch. Nearly All Binary Searches and Mergesorts are Broken, *Google*

Research Blog, February 2, 2006, `http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html`.

[7] J. M. Borwein, D. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*, A. K. Peters, Natick, MA, 2003.

[8] J. M. Borwein, D. Bailey, R. Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*, A. K. Peters, Natick, MA, 2004.

[9] A. Bundy, M. Jamnik, A. Fugard. What is a proof, *Phil. Trans. R. Soc.* A 363 (2005), 2377–2391.

[10] A. S. Calude. The journey of the four colour theorem through time, *The NZ Math. Magazine* 38, 3 (2001), 27–35.

[11] C. S. Calude. *Theories of Computational Complexity*, North-Holland, Amsterdam, 1988.

[12] C. S. Calude, Elena Calude, S. Marcus. Passages of proof, *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 84 (2004), 167–188.

[13] C. S. Calude, G. J. Chaitin. A dialogue on mathematics & physics, *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 90 (2006), 31–39.

[14] C. S. Calude, S. Marcus. Mathematical proofs at a crossroad? in J. Karhumäki, H. Maurer, G. Păun, G. Rozenberg (eds.). *Theory Is Forever*, Lectures Notes in Comput. Sci. 3113, Springer-Verlag, Berlin, 2004, 15–28.

[15] B. Cipra. Gaps in a sphere packing proof? *Science* 259 (1993), 895.

[16] G. J. Chaitin. Two philosophical applications of algorithmic information theory, in C. S. Calude, M. J. Dinneen, V. Vajnovszki (eds.). *Proceedings DMTCS'03*, Springer Lecture Notes in Computer Science, vol. 2731, 2003, 1–10.

[17] G. J. Chaitin. *From Philosophy to Program Size*, Tallinn Institute of Cybernetics, 2003.

[18] G. J. Chaitin. *Meta Math!*, Pantheon, 2005.

[19] G. J. Chaitin. Epistemology as information theory: from Leibniz to Omega, *Collapse* 1 (2006), 27–51.

[20] Clay Mathematics Institute: Millennium Prize Problems, 2000, `http://www.claymath.org/millennium/Poincare_Conjecture/`.

[21] R. A. De Millo, R. J. Lipton, A. J. Perlis. Social processes and proofs of theorems and programs, *Comm. ACM* 22, 5 (1979), 271–280.

[22] K. Devlin. *The Millennium Problems*, Basic Books, Cambridge, MA, 2002.

[23] K. Devlin. When is a proof, *Devlin's Angle*, `http://www.maa.org/devlin/devlin_06_03.html`.

[24] Edsger W. Dijkstra. Real mathematicians don't prove, *EWD1012*, 24 January 1988, `http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1012.html`.

[25] L. F. Fass. Approximations, anomalies and "Ther Proof of Correctness Wars", *ACM SIGSOFT* 29, 2 (2004), 1–4.

[26] J. H. Fetzer. Program verification: the very idea, *Comm. ACM* 31 (1988), 1048–1063.

[27] Flyspeck, `http://www.math.pitt.edu/~thales/flyspeck/` (consulted 28 March 2007)

[28] S. Garfinkel. History's worst software bugs, *Wired*, August 11,

2005, http://www.wired.com/news/technology/bugs/0,2924,69355,00.html?tw=wn_tophead_1.

[29] D. Goldston, J. Pintz, C. Yildirim. Primes in tuples, http://arxiv.org/math.NT/0508185.

[30] R. K. Guy. *Unsolved Problems in Number Theory*, Springer-Verlag, New York, 1994, 2nd ed. (pp. 19–23).

[31] T. C. Hales. Sphere packings. I, *Disc. Comput. Geom.* 17 (1997), 10–51.

[32] T. C. Hales. A proof of the Kepler conjecture, *Ann. Math.* 162 (2005), 1065–1185.

[33] D. Hilbert. Mathematical problems (Lecture delivered before the International Congress of Mathematicians at Paris in 1900), in F. E. Browder (ed.). *Proceedings of Symposia in Pure Mathematics of the AMS*, AMS, 28, 1976, p. 25.

[34] C. A. R. Hoare. How did software get so reliable without proof?, in *Proceedings of the Third International Symposium of Formal Methods Europe on Industrial Benefit and Advances in Formal Methods*, Lect. Notes Comput. Sci., Vol. 1051, Springer-Verlag London, 1996, 1 –17.

[35] D. Kennedy. Breakthrough of the year, December 22, 2006, http://www.sciencemag.org/cgi/reprint/314/5807/1841.pdf.

[36] D. E. Knuth. Theory and practice, *EATCS Bull.* 27 (1985), 14–21.

[37] D. Mackenzie. *Mechanizing Proof*, MIT Press, Cambridge, Mass. 2001.

[38] T. R. Nicely. Enumeration to 10^{14} of the twin primes and Brun's constant, *Virginia Journal of Science* 46, 3 (Fall, 1995), 195–204.

[39] G. Odifreddi. *The Mathematical Century*, Princeton University Press, Princeton, 2004.

[40] G. Perelman. Ricci Flow and Geometrization of Three-Manifolds, Massachusetts Institute of Technology, Department of Mathematics Simons Lecture Series, September 23, 2004 http://www-math.mit.edu/conferences/simons.

[41] G. Perelman. The Entropy Formula for the Ricci Flow and Its Geometric Application, November 11, 2002, http://www.arxiv.org/abs/math.DG/0211159.

[42] G. Perelman. Ricci Flow with Surgery on Three-Manifolds, March 10, 2003, http://www.arxiv.org/abs/math.DG/0303109.

[43] H. Poincaré. *Oeuvres de Henri Poincaré*, tome VI. Gauthier-Villars, Paris, 1953, pp. 486 and 498.

[44] K. R. Popper. Indeterminism in quantum physics and in classical physics. Part II, *The British Journal for the Philosophy of Science* 1, 3 (1950), 173–195.

[45] K. R. Popper. Part 1 in David Miller (ed.). *Popper Selections*, Princeton University Press, 1985.

[46] G.-C. Rota. *Indiscrete Thoughts*, Birkhäuser, Boston, 1997, p. 96.

[47] J.-P. Serre. Interview, *Liberation* May 23 (2003).

[48] J. Radhkrishnan, M. Sudan. On Dinur's proof of the PCP theorem, *Bull. AMS* 44, 1 (2007), 19–61.

[49] J. Schwartz. The pernicious influence of mathematics on science, in M. Kac,

G.-C. Rota, J. Schwartz (eds.). *Discrete Thoughts*, Birkhäuser, Boston, 2nd ed.,1992, 19–26.f

[50] M. Schwartz. Google Groups XSS Bug [Part 2], January 3, 2007, `http://weblogs.asp.net/mschwarz/`.

[51] S. Shapiro. Computability, proof, and open-texture, in Adam Olszewski, Jan Woleński, Robert Janusz (eds.). *Church's Thesis After 70 Years*, Ontos Verlag, Berlin, 2006, 420–455.

[52] K. Soundararajan. Small gaps between prime numbers: the work of Goldston-Pintz-Yildirim, *Bull. AMS* 44, 1 (2007), 1–18.

[53] R. Thomas. The Four Color Theorem, `http://www.math.gatech.edu/~thomas/FC/fourcolor.html`.

[54] W. P. Thurston. On proof and progress in mathematics, *Bull. AMS* 30, 2 (1994), 161–177.

[55] Eric W. Weisstein. Kepler Conjecture. From *MathWorld–A Wolfram Web Resource*, `http://mathworld.wolfram.com/KeplerConjecture.html`.

[56] D. Zeilberger. Theorems for a price: tomorrow's semi-rigorous mathematical culture, *Notices AMS* 40(1993), 978–981.

[57] D. Zeilberger. Don't Ask: What Can The Computer do for ME?, But Rather: What CAN I do for the COMPUTER?, March 5, 1999, `http://www.math.rutgers.edu/~zeilberg/Opinion36.html`.

[58] Asia's Net: Hanging by a Thread, January 4, 2007, `http://www.time.com/time/magazine/article/0,9171,501070115-1573933,00.html`.

[59] PayPal Security Flaw Allows Identity Theft, `http://news.netcraft.com/archives/2006/06/16/ paypal_security_flaw_allows_identity_theft.html`.

[60] Therac25: Computerized Radiation Therapy, `http://neptune.netcomp.monash.edu.au/ cpe9001/assets/readings/www_uguelph_ca_~tgallagh_~tgallagh.html`.

[61] New Year Bug Cripples Microsoft Expression Preview, January 3, 2007, `http://www.ddj.com/dept/debug/196800800`.

[62] Bad Software May Have Doomed Mars orbiter , `http://www.msnbc.msn.com/id/16580498`.

Chapter 19

God's Number: Where Can We Find the Secret of the Universe? In a Single Number!

Marcus Chown[1]

New Scientist, London, UK; MChown@compuserve.com

Computers are useless. They can only give you answers.
Pablo Picasso
There is hardly any paradox without utility.
Gottfried Leibniz

Some time ago a group of hyper-intelligent pan-dimensional beings decided to answer the great question of Life, The Universe and Everything. To this end they built an incredibly powerful computer, Deep Thought. After the great computer program had run for a few million years, the answer was announced. And the answer was ...

.0000001000000100001000001000011101110011001001111000100100111000 ...

Come again? Surely, it was 42? Well, in Douglas Adams' novel *The Hitch Hiker's Guide to the Galaxy* it certainly was. But, in the real world, rather than the world of Arthur Dent, Zaphod Beeblebrox and Ford Prefect, the answer to the question of Life, The Universe and Everything is very definitely...

.0000001000000100001000001000011101110011001001111000100100111000 ...

The number is called Omega and, remarkably, if you knew its first few thousand digits, you would know the answers to more mathematical questions than can ever be posed. What is more, the very existence of Omega is a demonstration that most mathematics cannot be discovered simply by

[1]This article was first published in the book Marcus Chown. *The Never-Ending Days of Being Dead* (Faber, London, 2007). It is re-published here with the kind permission of Faber.

applying logic and reasoning. The fact that mathematicians have little difficulty in discovering new mathematics may therefore mean that they are doing something—employing "intuition" perhaps—that no computer can do. It is tantalising evidence that the human brain is more than a jelly-and-water version of the PC on sitting on your desktop.

Omega (Ω) actually crops up in a field of mathematics invented by an Argentinian-American called Gregory Chaitin. Algorithmic Information Theory attempts to define "complexity".[2]

This is a very difficult concept to pin down precisely yet a precise definition is extremely important in many fields. How else, for instance, can a biologist studying evolution objectively say that a human is more complex than a chimpanzee or even a jellyfish?

Chaitin invented AIT when he was 15, the same age Wolfram was when he began publishing papers in physics journals. His principal concern at the time was with numbers. But, in fact, AIT applies to much more. After all, as we all know today, information describing everything—from words to pictures to music—can ultimately be expressed in the form of numbers. We are living in a "digital" world.

Chaitin's key idea was that the complexity of a number can be measured by the length of the shortest computer program that can generate it. Take, for instance, a number which goes on forever such as 919191... Although it contains an extremely large number of digits—it goes on forever after all—it can be generated by a very short program:

Step 1: Write "91"
Step 2: Repeat step 1

According to Chaitin's measure, therefore, the number 919191... is not very complex at all. The information it contains can be reduced, or "compressed", into a much more concise form than the number itself—specifically, the two-line program above.

Actually, Chaitin is a bit more precise about what he means by the "shortest computer program". He is a mathematician, after all. He means the "shortest computer program encoded in binary that can generate a particular number, itself expressed in binary ".

[2]Many of the ideas of AIT were invented independently by the Russian Andrei Kolmogorov.

Binary means a string of 0s and 1s and is pretty much synonymous with the word digital.[3] All computer programs—including Microsoft Windows—are ultimately encoded in binary. So it is not hard to imagine a computer program with strings of 0s and 1s representing both numbers and commands such as "Repeat step 1". It is the length of just such a program that Chaitin equates with the complexity of a number.

According to AIT, if there are two numbers and generating the first requires a program of length 37 binary digits, or bits, while generating the second requires one 25 bits long, the first number is the more complex.

Pattern is the key. If the digits of a number have some kind of pattern—like the pattern of 919191...—the pattern can be used as a shortcut to generate the number. Consequently, the binary program necessary to generate the number is relatively short—it has fewer bits than the number itself. Such a number is said to contain reducible information because it can be reduced, or compressed, into a more compact form—the form of the computer program.

Most numbers have no discernible pattern, however. Unlike 919191..., their digits are entirely unrelated to each other. The only way for a computer program to generate such a number is to write it out in full. This is no compression at all. The program is as long as the number itself. Such a number is said to contain irreducible information because it cannot be squeezed into a more compact form. This is where Omega, "the jewel in the crown of AIT", comes in. It takes the idea of irreducible information to its insane, logical extreme.

Omega, which was first defined by Chaitin in the 1970s, is an infinitely long number whose digits are without the slightest trace of pattern. Consequently, there is no way to generate its first 10 digits with a program less than 10 digits long; no way to create its first 511 digits with a program shorter than 511 digits in length; and so on. Omega's never-ending sequence of 0s and 1s can be generated only with an infinitely long computer

[3]Binary was invented by the 17th-century mathematician Gottfried Leibniz. It is a way of representing numbers as a strings of 0s and 1s. In everyday life, we use decimal, or base 10. The right-hand digit represents the 1s, the next digit the 10s, the next the 10×10s, and so on. For instance, 9217 means $7+1\times10+2\times(10\times10)+9\times(10\times10\times10)$. In binary, or base 2, the right-hand digit represents the 1s, the next digit the 2s, the next the 2×2s, and so on. So, for instance, 1101 means $1+0\times2+1\times(2\times2)+1\times(2\times2\times2)$, which in decimal is 13.

program. There is no shortcut, no way to compress it into a more compact form. It is the ultimate in irreducible information.[4]

Chaitin calls it a "very dangerous number". "On the one hand, it has a simple, straightforward mathematical definition," he says. "On the other hand, its actual numerical value is maximally unknowable." People often think of pi—the ratio of the circumference to a diameter of the circle—as complex. After all, its digits—3.415926...—go on forever and do not appear to repeat. However, it turns out that pi can be generated by a relatively simple computer program and so, by Chaitin's measure, is not very complex at all. "By comparison, Omega is infinitely more complex," says Chaitin.

Omega is like an infinite series of coin tosses—with the "heads" equivalent to 0s and the "tails" to 1s. The outcome of each toss in such a series is entirely unrelated to that of the previous toss. The only one way to discover the sequence of "heads" and "tails" in an infinite series of coin tosses is to toss a coin an infinite number of times. There is no shortcut. "And this is exactly the way it is with the digits of Omega," says Chaitin.

But Omega, it turns out, is more—much more—than simply an infinitely random, infinitely complex, infinitely incompressible number. Unexpectedly, it turns out to have a deep connection with the ultimate limitations of computers—what they can and cannot compute.

19.1. Uncomputability

The question of what computers can and cannot do was an obsession of the English mathematician Alan Turing. During the Second World War, Turing was stationed at Britain's top secret code-breaking establishment at Bletchley Park in Buckinghamshire. There he helped break the fiendishly complex "Enigma" and "Fish" codes with which the Nazis encoded their most secret radio transmissions. The intelligence gathered enabled Winston Churchill and the Allies to anticipate German actions, saving countless lives by shortening the war, some say by up to two years.

Turing's code-breaking success relied on "Colossus", the world's first programmable electronic computer, more than 10 of which were in oper-

[4]Actually, there is not one Omega but a whole class of Omegas. This is because Omega depends on the particular type of computer language used to generate a number. It would not be the same, for instance, in two languages that used a different string of 0s and 1s to code for a task like "Repeat step 1".

ation at Bletchley Park and Cheltenham by the end of the war. But his enduring fame rests on work he carried out earlier, in the 1930s, on a far more theoretical type of computer—one he invented with the specific purpose of figuring out the limits of computers.

A "Turing machine" is simply a box. A 1-dimensional tape with a series of 0s and 1s inscribed on it is fed into the box and the same tape emerges from the box with a different series of 0s and 1s on it. The "input" is transformed into the "output" by a read/write head in the box. As the tape passes the head, one digit at a time, the head either leaves the digit unchanged or erases it, replacing a 0 by a 1, and vice versa. Exactly what the head does to each digit is determined by the "internal state" of the box at the time—what, in today's jargon, we would call a computer program.

With its input and output written in binary on a 1-dimensional tape, a Turing machine is a wildly impractical device. Practicality, however, was not the point. The point was that, with the Turing machine, Turing had invented—on paper, at least—a machine that could simulate the operation of absolutely any other machine.[5]

Nowadays, a machine that can simulate any other machine—a "universal machine" —is not considered remarkable at all. Such devices—capable of carrying out not one specialised task but any conceivable task—are ubiquitous features of the world. They are called computers. In the 1930s, however, the universal Turing machine appeared to be straight from the pages of science fiction. The only way computing machines of the day could carry out different tasks was if they were painstakingly rewired. Turing's genius was to see that this was unnecessary. With a universal machine—a general-purpose computer—it was possible to simulate any other machine simply by giving it a description of the other machine plus the computer program for the other machine. There was no need to change the wiring—the hardware—only the software.

Turing imagined the software for his universal Turing machine inscribed as a long series of 0s and 1s on a cumbersome 1-dimensional tape. Fortunately, today's computers are a bit more sophisticated than Turing's vision and their software comes in a considerably more user-friendly form!

In the universal Turing machine, however, Alan Turing can in fact be

[5]And it could do this using only seven basic commands! (i) Read tape, (ii) Move tape left, (iii) Move tape right, (iv) Write 0 on tape, (v) Write 1 on tape, (vi) Jump to another command, and (vii) Halt.

said to have invented the modern general-purpose computer, a machine whose unprecedented flexibility is guaranteed by infinitely re-writable software.

Turing's genius was to recognise that, in the final analysis, all a computer really is is a device for shuffling symbols. One sequences of symbols is fed in. And another sequence of symbols is spat out which depends on its computer program. This is all any computer does. Take, for instance, the computer that flies a plane. It is fed sequences of symbols which tell it the position of the plane, its speed, the engine temperature, and so on. The program then acts on this information, telling the computer what sequences of symbols it should spit out in order to control engine revs, rudder direction and so on.

A universal Turing machine similarly was a symbol shuffler. But the key thing for Turing was that it could simulate the operation of any conceivable machine. This meant that it could compute anything that was computable, and, recall, he was interested in what could and could not be computed. All he had to do, therefore, was find a task that would flummox a universal Turing machine. Remarkably, he stumbled on one immediately. Even more remarkably, the task was fantastically simple.

The impossible task was to take a computer program—any computer program—and determine whether or not it ever stops.

What exactly does this mean? Well, people who write computer programs for a living know that such programs sometimes get stuck in endless loops, going round and round the same set of instructions like a demented hamster in a wheel. The task Turing set his universal Turing machine was to take a program and compute whether or not this will happen—Spitting out, say a 0 if the answer is, yes, and a 1 if it is, no.

At first sight, this "halting problem" seems ridiculously simple to solve. The easiest way to check whether a particular program halts or not is simply to run it on a computer and see. This is certainly feasible if the program comes to a halt after a minute or an hour or even after a year. But what if it halts only after 1000 years or a trillion trillion years? Nobody can wait that long. Now, perhaps, the deceptive trickiness of the halting problem is apparent. It is about taking a program and computing *in advance of actually running it* whether it will "eventually" stop.

And the remarkable thing that Turing discovered was that this apparently simple task is impossible. Though it is easy to state, no conceivable

computer, no matter how powerful, can ever compute it.[6]

The halting problem would appear to have nothing whatsoever to do with Omega (though, like the halting problem, Omega is simple to define but impossible to compute). However, the two are very deeply connected. Omega, it turns out, is more than irreducible information, naked randomness. Omega is the "probability" that a randomly chosen computer program—one picked blindly from all possible computer programs—will eventually halt.

Probabilities are conventionally written as fractions between 0 and 1, with a probability of 0.5 corresponding to a 50 per cent chance of something happening, a probability of 0.99 to 99 per cent and so on. Since Omega is defined as a probability, there is in fact a decimal point before its first digit, as can be seen in the Omega written out at the beginning of this chapter.

But what does it mean that Omega is the probability that a randomly chosen computer program halts? Well, think of generating a string of 0s and 1s by repeatedly tossing a coin. Such a string can encode the instructions of a computer program just as easily as it can encode the bars of a piece of music or the picture elements of a family photo. Well, Omega is the probability that a computer program generated in a random manner like this will eventually halt.

Put it another way. Omega is what you get when take all possible computer programs that can exist, one at a time. You see whether or not each halts, giving a 0 for one possibility and a 1 for the other. Because there are an infinite number of possible programs, there will be an infinite number of 0s and 1s. Well, you take the average of all the 0s and 1s. And that is Omega.

---

[6]The halting problem is uncomputable because, if there was a program that could compute it—one that could take another program and spit out, say, a 0 if it never halts and a 1 if it eventually halts—this "halting program" could be used to do something impossible: construct a program which stops if it doesn't stop and doesn't stops if it stops!

How? Such a program would have to incorporate the halting program as sub-program, or subroutine, and apply it to itself. This sounds tricky but actually is not. Just engineer the program to output itself—a string of 0s and 1s identical to the binary code of the program—and then get the halting program to check whether the output halts or not. If it does halt, the program has instructions not to halt; and if it does not halt, the program has instructions to halt.

What has been concocted is an impossibility, a contradiction—all made possible by the existence of the halting program. For the sanity of the Universe, therefore, the halting program cannot exist.

Put it another way. Omega is the concentrated distillation of all conceivable halting problems. It contains the answer not to just one halting problems but to an infinite number!

Of course, individual cases of the halting problem are uncomputable. This was Turing's big discovery, after all. Consequently, Omega too is uncomputable. This is in fact not very surprising. Recall that it takes an infinitely long computer program to generate Omega, which is hardly a practical proposition!

Omega is maximally uncomputable, maximally unknowable. "Technically, this is because the first n bits would in theory enable us to answer Turing's infamous halting problem for all programs up to n bits in size—and this is impossible," says Chaitin. "However, crudely speaking, the reason Omega is unknowable is that it's the probability of something happening—a computer program halting—which itself is unknowable!"

The deep an unexpected connection between Omega and all conceivable halting problems has an astonishing consequence. "It comes about because of the remarkable fact that most of the interesting problems in mathematics can be written as halting problems," says Cristian Calude of the University of Auckland.

Take, for example, the problem of finding a whole number that is not the sum of three square numbers. The number 6, for instance, fails. It can be written as $1^2 + 2^2 + 1^2$ and so is the sum of three square numbers. The first number that is not a sum of three squares is in fact 7.

A brute force program to find numbers that are not the sum of three squares would simply step through the whole numbers, one at a time, stopping when it finds a number that cannot be written as the sum of three squares. Or, if all numbers can be written as the sum of three squares, it will keep going forever. "Does this ring any bells?" says Calude. "It's a halting problem!"

The amazing thing is that a host of other mathematical questions can also be re-cast as halting problems—if a particular program halts, the answer to the question is, yes; if it doesn't halt, it is no. The consequence of this fact is scarcely believable. "The first few thousand digits of Omega contain the answers to more mathematical questions than could be written down in the entire universe!" says Charles Bennett of IBM in New York.

An example of such a question is whether "Fermat's last theorem" is

correct. This was inserted in the margin of a book by Pierre de Fermat in 1642 and finally proved three centuries later by Andrew Wiles in 1995. It asserts that there no positive whole numbers $x^n + y^n = z^n$, if n is bigger than 2. There are, for instance, no whole numbers $x^3 + y^3 = z^3$ or $x^{99} + y^{99} = z^{99}$. Another example of a mathematical question is whether the "Goldbach conjecture" is correct. This states that every even number greater than 2 is the sum of two prime numbers (A prime number being a number, like 3 or 111, which is divisible only by itself and 1). Though the conjecture was stated by Goldbach in a letter to the great Swiss mathematician Leonhard Euler in 1742, it has defied all attempts of mathematicians to prove it right or wrong.

The question of whether the Goldbach conjecture is correct can be couched as a halting problem. So too can the matter of whether or Fermat's last theorem is right. "To solve these important problems—and many other—it is therefore necessary only to know the bits of Omega!" says Calude.

Omega, it appears, is so much more than a maximally unknowable, maximally uncomputable, maximally random number. It is so much more than the distillation of all conceivable halting problems. As Chaitin puts it: "Omega is also the diamond-hard distilled and crystallised essence of mathematical truth."

Bennett is even more lyrical. "Throughout history mystics and philosophers have sought a compact key to universal wisdom, a finite formula or text which, when known and understood, would provide the answer to every question," he says. "The Bible, the Koran, the I Ching and the medieval Jewish Cabala exemplify this hope. Omega is in many senses a cabalistic number. It can be known of, but not known through human reason. To know it in detail, one would have to accept its uncomputable digit sequence on faith, like words of a sacred text."

There you have it. Omega is a "compact key to universal wisdom". It provides the answer to every question—at least, every mathematical question.

John Casti of the Technical University of Vienna goes one step farther. "Omega's digits encode 'the secret of the universe'," he says. "Almost every unsolved problem in mathematics and many in physics and elsewhere could be settled by knowing enough digits of Omega." [7]

[7]Actually, the early 20th-century French mathematician Émil Borel was the first to

Of course, Omega may contain the "secret of the universe" but it is unknowable. In fact, it is worse than this. "Even if, by some kind of miracle, we get the first 10,000 digits of Omega, the task of solving the problems whose answers are embodied in these bits is computable but unrealistically difficult," says Calude. "Technically, the time it takes to find all halting programs of length less than n grows faster than any computable function of n."

In other words, we will be in the position of the characters in Adams' *The Hitch Hiker's Guide to the Galaxy.* They knew the answer to Life, the Universe and Everything is 42. Unfortunately, the hard part was knowing the question.

Determining all the digits of Omega is clearly an impossibility for lowly human beings. The number in its entirety is really knowable only by God. Incredibly, however, Calude has managed to calculate the first 64 digits of Omega—or at least *an* Omega. Those digits are the ones shown at the beginning of this chapter.

Calude was able to calculate 64 bits of a nominally uncomputable number because, contrary to everything that has been said up to now, the computing barrier discovered by Turing can actually be broken. This is because Turing defined the halting problem for a classical Turing machine—a familiar general-purpose computer. However, nature permits types of machines that Turing did not anticipate such as "quantum computers". These are "accelerated Turing machines". It may be possible to use them solve the halting problem and compute other apparently uncomputable things.

---

show how a number could encapsulate the answers to all conceivable questions. Take the French alphabet, he said, including blanks, digits, punctuation marks, upper case, lower case, letters with accents, and everything. Then start making a list. Start off with all possible 1-character sequences, then all possible 2-character sequences, and so on. Eventually, you will get all possible successions of characters of any length, in alphabetical order. Most will of course be nonsense. Nevertheless, in the list, you will find every conceivable question in French—in fact, everything you can write in French. Next, said Borel, number the sequences you have created. Then imagine a number $0.d_1d_2d_3\ldots$ whose nth digit d_n is a 1 if the nth element of the list is a valid yes/no question in French whose answer is, yes, and whose nth digit is 2 if the nth element is a valid yes/no question whose answer is, no. If the nth element of the list is garbage, not valid French, or valid French but not a yes/no question, then the nth digit is 0.
So Borel had a number that gives the answer to every yes/no question you can ask in French—about religion, about maths, about physics—and it is all in one number! Of course, such a number would contain an infinite amount of information, which would make actually ever knowing it a bit unrealistic. It would be just like Omega. In fact, Borel's number is actually related to Omega.

Calude himself used ingenious mathematical tricks to partially circumvent the Turing barrier and compute 64 bits of Omega—a feat even Chaitin, the inventor of Omega, had believed impossible.

Calude's demonstration that it is possible to know at least some of the digits of Omega has left a strong impression on Chaitin. He has gone so far as to suggest that knowledge of Omega could be used to characterise the level of development of human civilisation. Chaitin points out that, in the 17th-century, the mathematician Gottfried Leibniz observed that, at any particular time, we may know all the interesting mathematical theorems with proofs of up to any given size, and that this knowledge could be used to measure human progress. "Instead, I would propose that human progress—purely intellectual not moral—be measured by the number of bits of Omega that we have been able to determine," says Chaitin.

"What you don't know is also a kind of knowledge," said Jostein Gaarder in *Sophie's World*.

Calude has even suggested that, if we wish to signal our existence to the stars, the way to impress an extraterrestrial civilisation and show ourselves worthy of contact may be to broadcast as many digits as we can of Omega.

But Omega, in addition to being related to what computers can and cannot do, and distilling the answers to all mathematical questions, has yet more miraculous properties (There appear to be no end to them!). To understand what those properties are, it is necessary to appreciate a major crisis which occurred in mathematics in the late 19th century. The crisis was triggered by a field of mathematics called "set theory".

19.2. Undecidability

Set theory is concerned with groups of objects known—not surprisingly—as "sets". Examples include the set of all countries beginning with the letter "A" ; the set of all odd numbers; and the set of all mammals. Sets can be related to each other. For instance, one set can be contained within another set. The set of all marsupials, for instance, is a "subset" of the set of all mammals which, in turn, is a subset of the set of all animals.

Although the basic idea of a set is straightforward, it turns out that set theory permits the existence of a particularly catastrophic set—the set of all sets that are not members of themselves. Why is this catastrophic? Well, try asking whether this set is a member of itself? Immediately, it is

apparent that the set is a member of itself only if it is not a member of itself! "It's like the paradox of the village barber who shaves every man in a village who doesn't shave himself," says Chaitin. "Who shaves the barber? He shaves himself if and only if he doesn't shave himself. It is a contradiction."

This set paradox is closely related to an even older paradox: the declaration by the Greek philosopher Epimenides "This statement is false!". Since the statement is true only if it is false, it is neither true nor false. And Epimenides' paradox is in turn just another form of the "liar's paradox" : the assertion by someone: "I am a liar."

For mathematicians of the late 19th century the contradiction in set theory was the stuff of nightmares. The very foundation of mathematics was logical reasoning. Yet here was a case where logical reasoning led to an absurdity. Mathematics was widely regarded as a realm of pure, clear-cut truths, a lofty ethereal domain far removed from the messiness of the everyday world. After all, it dealt with things which were true, demonstrably and beyond any possible doubt, not simply at this moment but throughout all eternity. Yet now the mathematicians' sanctuary of beauty and perfection was under mortal threat. At all costs the contradiction in set theory must be eradicated. And the man took on the task of eradicating it was the greatest mathematician of his day, the German David Hilbert.

There are thousands of fields of mathematics, many of which are interconnected. However, each has the same basic structure. On top of a bedrock of "axioms" mathematicians erect a scaffolding of "theorems". The axioms are self-evident truths, simple assertions on which all mathematicians can agree (no progress in mathematics is possible without assuming some things). The theorems are the logical consequences of the axioms. For instance, Euclidian, or flat-paper, geometry consists of a handful of axioms about straight lines and the angles between them—for example, "parallel lines never meet"—plus the theorems that can be deduced from such axioms—for example, "the internal angles of a triangle always add up to 180 degrees".

Hilbert's big idea was to first identify a small group of axioms as the bedrock of *all* of mathematics. Once this was done, the next step was to spell out in gory, painstaking detail the logical rules for getting from the axioms to theorems (or vice versa). This would make it possible to "prove" any mathematical statement—that is, show that it can be obtained by log-

ical steps from the bedrock axioms and so is a bona fide theorem.

In short, what Hilbert had in mind was finding a proof-checking "algorithm" —a procedure for checking that each step in a given proof is logically watertight. If mathematicians possessed such a procedure they would in theory be able to run through all possible proofs, starting with the simple ones and progressing to more complex ones; check whether they are correct; and see what theorems they led to. In this way they would be able to generate an infinite list of all provable mathematical statements—that is, all theorems.

If a mathematical statement is true, Hilbert's mindless approach would therefore eventually find the proof. If a statement cannot be proved, Hilbert's mindless method would go on forever, unless a proof that the statement is false was found.

The mechanical nature of Hilbert's proof-checking procedure was crucially important. After all, if it could be applied mindlessly, without any need to know how mathematics worked, then it would be something absolutely everyone could agree on. Hilbert would have taken the process of doing mathematics and set it in stone. He would have removed from the subject all the ambiguities of everyday language and reasoning. There would be no room left for contradictions such as the one that appeared to have cropped up in set theory.

Hilbert did not know it—could not have known it—but the mechanical proof-checking procedure he envisaged was nothing less than a computer program running on a computer! "How many people realise that the computer was actually conceived and born in the abstract field of pure mathematics?" says Chaitin.

Hilbert's programme to weed out the paradoxes from mathematics was hugely ambitious. He fully expected it to take decades to carry out. But what he did not realise—and nor did anyone else—was that the programme was impossible!

In 1931, an obscure Austrian mathematician called Kurt Gödel showed that, no matter what set of axioms you select as the ultimate bedrock of all mathematics, there will always be theorems—perfectly legitimate theorems—that you can never deduce from the axioms. Contrary to all expectations, the perfect world of mathematics is plagued by "undecidable" theorems—things which are true but which can never be proved to be true by logical, rational reasoning.

Gödel proved his result in the most ingenious way. He managed to embed in arithmetic—one of the most basic fields of mathematics—the self-referential declaration "this statement is unprovable". Since this required him to make a piece of arithmetic actually refer to itself, it was an immensely difficult task. However, by embedding the troublesome statement in arithmetic, Gödel had buried an atomic bomb in the very fabric of mathematics. "This statement is unprovable" is, after all, the "liar's paradox" in another guise. If it is true, mathematics admits false statements that cannot be proved—it is inconsistent. If it is false, it admits undecidable statements that can never be settled—it is incomplete.

Incompleteness is very bad for mathematics but inconsistency is truly terrible. False statements would be like a plague of moths gnawing at its very fabric. There was no choice for mathematicians but to accept the lesser of Gödel's two evils. Mathematics must be terminally incomplete. To everyone's profound shock, it contained theorems that could never be proved to be true.

"All theorems rest on premises," declared Aristotle. Gödel's "incompleteness theorem" shows that the great man was sorely mistaken. High above the mathematical bedrock there are pieces of mathematical scaffolding floating impossibly in mid-air.

The obvious way to reach these free-floating theorems is by building up the bedrock—that is, adding more axioms. However, this will not help. According to Gödel's incompleteness theorem, no matter how many axioms are added, there will always be theorems floating in the sky, perpetually out of reach. There will always be theorems that are true but that can never be proved to be true, at least by logical, rational reasoning.

To say that Gödel's discovery was deeply distressing to mathematicians is a bit of an understatement. As pointed out, mathematicians had believed mathematics was a realm of certain truths, far from the messy uncertainty of the everyday life. This is precisely what had attracted many of them to the field in the first place. But, contrary to expectations, mathematics turned out to be a realm where many things are up in the air, many things are messy, many things are uncertain[8] Some mathematician could not hide

[8]By an odd coincidence, physicists were having a similarly shocking experience at about the same time. The microscopic realm of atoms, they discovered, was a place of unpredictability and uncertainty. Not only do things such as the disintegration of an atom occur for no reason at all, it is not even possible to be 100 per cent certain of basic things

their despair at this unhappy revelation. "Gödel's result has been a constant drain on my enthusiasm and determination," wrote Hermann Weyl.

No matter how unpalatable it might be, however, Gödel's result was incontrovertible. Mathematicians had no choice but get used to it—even to revere it. Many now consider the publication of Gödel's incompleteness theorem to be the most significant event in the whole of 20th century mathematics. "Gödel's incompleteness theorem has the same scientific status as Einstein's principle of relativity, Heisenberg's uncertainty principle and Watson and Crick's double helix model of DNA," says Calude.

But, if things were bad in the world of mathematics after Gödel discovered incompleteness, they got a whole lot worse five years later. That was when Turing discovered uncomputability[9] Not only did mathematics contain things that were undecidable, it also contained things such as the halting problem that were uncomputable.

Undecidability is in fact deeply connected to the uncomputability[10] Not only that but both undecidability and uncomputability are also deeply connected to Chaitin's idea that the complexity of a number is synonymous with the shortest program that can generate the number.

This is not obvious at all. However, recall that Omega is the ultimate in irreducible information. This means it cannot be generated by a program shorter than itself, which is the same as saying it cannot be compressed into a shorter string of bits than itself. Now think of one of those free-floating theorems that Gödel discovered are an inevitable feature of mathematics. It cannot be reached by logical deduction from any axioms, which is the same as saying it cannot be deduced from any principles simpler than itself,

---

like the position and speed of atoms. This uncertainty was quantified in "Heisenberg's uncertainty principle". If Gödel dropped a bombshell in mathematics, Heisenberg can be said to have dropped one in physics.

[9]The 29 March 1999 issue of *Time* magazine included both Gödel and Turing among the their 20 greatest scientists and thinkers of the 20th century.

[10]In fact, undecidability is a consequence of uncomputability. If it were always possible to start with some axioms and "prove" that a given program halts or that it never does, that will give you a way to "compute" in advance whether a program halts or not, How? You simply run through all possible proofs, starting with the simplest ones, checking which ones are correct, until either you find a proof the program will halt eventually or you find a proof that it is never going to halt. Since Turing showed that computing in advance whether or not a program will halt is impossible, it follows that this procedure too is impossible. It follows that there must be proofs—such as the proof that a given program will halt—that cannot be found by this logical, step-by-step process. In other words, there are proofs that cannot be deduced from any conceivable axioms and mathematics is incomplete.

compressed into any set of axioms. See the parallel?

19.3. Compressibility and what scientists do

Here is another interesting thing. Reducing theorems to a small number of axioms turns out to be deeply reminiscent of what scientists do. The mark of a good scientific theory, after all, is that it describes a large number of observations of the world while making only a small number of assumptions. In the words of the Nobel-prize-winning American physicist Richard Feynman: "When you get it right, it is obvious it is right—at least if you have any experience—because usually what happens is that more comes out than goes in."

Chaitin, as the inventor of AIT, has a unique take on this idea. "A scientific theory is really just a computer program that calculates observations," he says. "The smaller and more concise the computer program the better the theory."

Scientists have long known that, if there are two competing theories, both of which explain the same phenomenon, the one that makes the least assumptions is invariably the true one. This rule of thumb, known as "Ockham's razor", was first noted by William of Ockham, a Franciscan friar living in the 14th century.

By the criterion of Okham's razor, for instance, Creationism is inferior to the scientific view of the origin of the Universe because it requires many more assumptions. What you get out is not much better than what you put in. As Leibniz observed more than three centuries ago, a theory is convincing only to the extent that it is substantially simpler than the facts it attempts to explain.

According to Chaitin, AIT puts Ockham's razor on a precise footing for the first time. "Understanding is compression," says Chaitin. "Okham's razor is simply saying that the best scientific theory is the most compressible." He amplifies this. "A concise computer program for calculating something is like an elegant theory that explains some phenomenon," says Chaitin. "And if no concise theory exists, the phenomenon has no explanation, no pattern, there is nothing interesting about it—it is just what it is, that's all."

In physics, the Holy Grail is a "Theory of Everything", which distils all the fundamental features of reality into one simple set of equations that

could be written on the back of a postage stamp. "From the point of view of AIT, the search for the Theory of Everything is the quest for an ultimate compression of the world," says Chaitin.

"The most incomprehensible thing about the universe," Einstein famously said, "is that it is comprehensible." Chaitin, who equates comprehension with compression, would rephrase this. The most incomprehensible thing about the Universe is that it is compressible. This feature of the world is the reason we have been able to divine universal laws of nature, which apply in all places and at all times—laws which have enabled us to build computers and nuclear reactors and gain some degree of mastery over nature.

To Chaitin the compressibility of the Universe is a wonder. "For some reason, God employed the least amount of stuff to build the world," he says. "For some reason, the laws of physics are as simple and beautiful as they can be and allow us, intelligent beings, to evolve." This is a modern version of something noted by Leibniz: "God has chosen the most perfect world", he wrote. "The one which is the most simple in hypotheses and the most rich in phenomena."

Though we do not know why the laws underpinning the Universe are simple, the faith that they are is a powerful driving force of science. According to Feynman: "It is possible to now when you are right way ahead of checking all the consequences. Truth is recognisable by its beauty and simplicity."

19.4. Randomness

Back to Gödel. Although he had shocked and depressed mathematicians by showing that mathematics contains theorems which are undecidable, surprisingly his result did not make any difference to the day-to-day doing of mathematics. Weyl's pessimism was misplaced. "Mathematicians, in their everyday work, simply do not come across results that state that they themselves are unprovable," says Chaitin. "Consequently, the places in mathematics where you get into trouble seem too remote, too strange, too atypical to matter."

A more serious worry was Turing's demonstration that there are things in the world which were completely uncomputable. This is a very concrete result. It refers, after all, to computer programs, which actually calculate

things. On the other hand, the program Turing considered merely tried to figure out whether another program halts or not. It is hardly typical of today's computer programs, which carry out word processing or surf the Internet. Not surprisingly, therefore, none of these programs turn out to be undermined in any discernible way by the uncomputability of the halting problem.

It would seem that uncomputability and undecidability are too esoteric to bother about, that they can be swept under the carpet and safely forgotten about. This is indeed how it appeared for a long while. All was tranquil and quiet in the garden of pure mathematics. But then the gate squeaked on its rusty hinges and in walked Chaitin.

From the time he had been a teenager, Chaitin had been convinced that Gödel and Turing's results had far more serious implications for mathematics than anyone guessed. And he had resolved to find out what those implications were. It was this quest that had led him to invent AIT.

AIT is of course founded on the idea that the complexity, or information content, of a number is synonymous with the shortest computer program that can generate the number. However, at the core of AIT—just like at the core of Turing and Gödel's work—there is a paradox. It is actually impossible to ever be sure you have found the shortest possible program![11]

A shortest program, of course, exists. But this is not the point. The point is that, although it exists, you can never be sure you have found it. Determining whether you have turns out to be an uncomputable problem.

AIT is founded on uncomputability. The whole field is as riddled with holes as a swiss cheese. Uncomputability in fact follows from AIT[12]

And so does Gödel's incompleteness theorem. This turns out to be

[11] Say, there is a program that can decide whether a given program, p, is the shortest possible program capable of producing a given output. Now consider a program, P, whose output is the output of the smallest program, p, bigger than P that is capable of producing the given output. But P is too small a program to produce the same output as p. There is a contradiction! Therefore, an algorithm for deciding if a program p is as small as possible cannot exist.

[12] If you could always decide in advance whether a program halts or not, you could systematically check whether each small program halts or not, and if it does halt, run it and see what it computes, until you find the smallest program that computes a given number. But this would contradict Chaitin's result that you cannot ever be sure you have the smallest program for generating a given number. Consequently, there can be no general solution to the halting problem. It is uncomputable.

equivalent to the fact that it is impossible to prove that a sequence of digits is incompressible—that is, the shortest program has been found. "Everywhere you turn in my theory you find incompleteness," says Chaitin. "Why? Because the very first question you ask in my theory gets you into trouble. You measure the complexity of something by the size of the smallest computer program for calculating it. But how can you be sure that what you have is the smallest computer program possible? The answer is that you can't!"

In Chaitin's AIT, undecidability and uncomputability take centre stage. Most mathematical problems turn out to be uncomputable. Most mathematical questions are not, even in principle, decidable. "Incompleteness doesn't just happen in very unusual, pathological circumstances, as many people believed," says Chaitin. "My discovery is that its tendrils are everywhere."

In mathematics, the usual assumption is that, if something is true, it is true for a reason. The reason something is true is called a proof, and the object of mathematics is to find proofs, to find the reason things are true. But the bits of Omega—AIT's crowning jewel—are random. Omega cannot be reduced to anything smaller than itself. Its 0s and 1s are like mathematical theorems that cannot be reduced or compressed down to simpler axioms. They are like bits of scaffolding floating in mid-air high above the axiomatic bedrock. They are like theorems which are true for no reason, true entirely by accident. They are random truths. "I have shown that God not only plays dice in physics but even in pure mathematics!" says Chaitin.[13]

Chaitin has shown that Gödel and Turing's results were just the tip of the iceberg. Most of mathematics is composed of random truths. "In a nutshell, Gödel discovered incompleteness, Turing discovered uncomputability, and I discovered randomness—that's the amazing fact that some mathematical statements are true for no reason, they're true by accident," says Chaitin.

Randomness is the key new idea. "Randomness is where reason stops, it's a statement that things are accidental, meaningless, unpredictable and happen for no reason," says Chaitin.

[13]This is a reference to Einstein. Appalled by quantum theory, which maintained that the world of atoms was ruled by random chance, he said: "God does not play dice with the universe." Unfortunately, he was wrong! As Stephen Hawking has wryly pointed out: "Not only does God play dice, he throws them where we cannot see them."

Chaitin has even found places where randomness crops up in the very foundation of pure mathematics—"number theory". "If randomness is even in something as basic as number theory, where else is it?" says Chaitin. "My hunch is it's everywhere."

Chaitin sees the mathematics which mathematicians have discovered so far as confined to a chain of small islands. On each of the islands are provable truths, the things which are true for a reason. For instance, on one island there are algebraic truths and arithmetic truths and calculus. And everything on each island is connected to everything else by threads of logic so it is possible to get from one thing to another simply by applying reason. However, the island chain is lost in an unimaginably vast ocean. The ocean is filled with random truths, theorems disconnected forever from everything else, tiny "atoms" of mathematical truth.

Chaitin thinks that the Goldbach conjecture, which has stubbornly defied all attempts to prove it true or false, may be just such a random truth. We just happened to have stumbled on it by accident. If he is right, it will never be proved right or wrong. There will be no way to deduce it from any conceivable set of axioms. Sooner or later, in fact, the Goldbach conjecture will have to be accepted as a shiny new axiom in its own right, a tiny atom plucked from the vast ocean of random truths.

In this context, Calude asks an intriguing question: "Is the existence of God an axiom or a theorem?" !

Chaitin is saying that the mathematical universe has infinite complexity and is therefore not fully comprehensible to human beings. "There's this vast world of mathematical truth out there—an infinite amount of information—but any given set of axioms only captures a tiny, finite amount of this information," says Chaitin. "This, in a nutshell, is why Gödel's incompleteness is natural and inevitable rather than mysterious and complicated."

Not surprisingly, the idea that, in some areas of mathematics, mathematical truth is completely random, unstructured, patternless and incomprehensible, is deeply upsetting to mathematicians. Some might close their eyes, view randomness as a cancer eating away at mathematics which they would rather not look at but Chaitin thinks that it is about time people opened their eyes. And rather than seeing it as bad, he sees it as good. "Randomness is the real foundation of mathematics," he says. "It is a rev-

olutionary change in our worldview."

This is explosive stuff. But Chaitin is able to risk the ire of the mathematical community because he is an outsider. He works for IBM at its Thomas J. Watson Research Center in Yorktown Heights, New York. In fact, he helped to develop the company's influential "Unix" work station, the IBM RS/6000. Chaitin does not believe it is possible to break the mould from within mathematics. "To be a revolutionary it may be necessary to be on the outside," he says.

Crucially, the language of physics turns out to be mathematics. The equations which describe things such as the motion of baseballs through the air and planets orbiting the Sun are mathematical equations. Many have remarked on the unreasonable effectiveness of mathematics in physics. But, if most truths in mathematics are random truths, things which are true for no reason at all, what does this say about truths in physics? "Ultimately, can the universe be comprehended—the physical universe as well as the universe of mathematical experience?" says Chaitin.

It all depends on whether the physical universe, like the mathematical universe, is infinitely complex. If, as most physicists believe, the world of atoms is ruled by naked chance, the Universe does indeed contain randomness. Consequently, it is infinitely complex and unknowable in its entirety by human beings.[14]

What this means is that physicists like Stephen Hawking, who fully expect the discovery of a Theory of Everything in the next decade or so, are destined to be disappointed. Though we may acquire such a theory, we will never know for sure whether we have the ultimate Theory of Everything. We will never be able to prove the compression to be the ultimate one. There will always be the possibility that there might be a yet deeper and simpler theory with the same explanatory power, out there waiting to be found. As the American physicist John Wheeler, famous for coining the term "black hole", has pointed out: "Even if physicists one day get their hands on a Theory of Everything, they will still face the unanswerable question: why does nature obey this set of equations and not another?"

[14] It is possible, however, that the randomness in the Universe is only pseudorandomness. This is the controversial view of Stephen Wolfram (see chapter "Cosmic Computer"). He believes that the consequences of the simple laws of physics are extremely complicated and so the Universe is only simulating randomness. If he is right, then the Universe is ultimately comprehensible. It would not follow that, because the laws of physics are mathematical and mathematics is incomplete, that physics is incomplete too.

The Anglo-American physicist Freeman Dyson sees the impossibility of finding a Theory of Everything as a good thing. Unlike the pessimistic Weyl, who described incompleteness as a constant drain on his enthusiasm, Dyson sees it as an insurance policy that science will go on forever. Though a Theory of Everything may be elusive, the mundane, day-to-day practice of doing physics will go on. The discoveries of Gödel and Turing do not appear to limit them in any way. As Chaitin points out, we have no trouble building and operating computers, far and away the most complicated machines ever constructed. "Here we have a case where the physical behaviour of a big chunk of the universe is very understandable and very predictable and follows definite laws," says Chaitin.

Chaitin's insights raise a fascinating and intriguing question about how exactly mathematicians actually do mathematics, how they find new theorems. While step-by-step reasoning and logic enables them to from one idea to another idea within an island in the great ocean of mathematical theorems, it does not allow them to get from island to island. But this is something they emphatically do do.

Reason and logic is insufficient. Chaitin therefore thinks that mathematicians discover mathematics using insights which go beyond reason and logic. He thinks they use the kind of flashes of inspiration and intuition artists talk about. "Mathematics isn't about the consequences of rules, it's about creativity and imagination," says Chaitin. "For this reason it is possible to argue that the incompleteness theorems does not limit what mathematicians do."

"Mathematical proof is an essentially creative activity," said Émil Post, who came up with the idea of a Turing machine independently of Turing. The brain is doing something more than any mere computer. Many people have suggested this before. "You can know much more than you can ever prove," said Feynman. But who would have thought the idea would receive support from a field as abstract and esoteric as pure mathematics?

Chapter 20

Omega Numbers

Jean-Paul Delahaye[1]

Laboratoire d'Informatique Fondamentale de Lille
Centre National de la Recherche Scientifique (CNRS)
Université des Sciences et Technologies de Lille, France;
delahaye@lifl.fr

Omega numbers are disconcerting: they are both well defined and uncomputable. Yet the closer you look, the more remarkable they appear.

A little more than 20 years ago—in 1979—Martin Gardner and Charles Bennett published an article on a new number whose properties were so peculiar that it was considered a paradox. Discovered by Gregory Chaitin, the number was called Omega and denoted Ω. Up to then, the symbol Ω had been used in mathematics for a variety of purposes. But increasingly it was reserved for Chaitin's number, just as π came exclusively to represent Archimedes' constant at the beginning of the 18th century.

Ω belongs to an infinite family of numbers, that is, Chaitin's omega numbers. These numbers are perplexing, for each represents an improbable assortment of strangenesses. A subclass of Chaitin's omega numbers, the Solovay omega numbers, complicate the picture further. These classes of curious numbers are as important as rational, algebraic, or transcendantal numbers. But they are exceedingly abstract, bordering on the absurd; indeed, they call into question the nature of mathematical knowledge.

[1]This paper has first appeared in French in *Pour la Science*, Mai 2002, n0 295, pp. 98–103. Re-published with the kind permission of *Pour la Science*.

20.1. Computable numbers

Let us proceed step by step through the universe of real numbers, to examine the definition and properties of omega numbers and to get a feeling for all their eccentricities.

Real numbers are numbers that, written in base 10, for example, can continue indefinitely (such as $e = 2.7182818284590\ldots$). Those that cannot continue indefinitely (like the famous 6.55957—the value of a euro in French francs at the launch of the single currency in 2002) are decimal fractions. Of those that do go on indefinitely, some do it in a periodic manner, for instance, $24/110 = 0.21818181818\ldots$ In addition, numbers that become periodic at a certain point in their expansion are the quotients of two whole numbers (called rational numbers). Irrational numbers such as the $\sqrt{2}$ are not quotients of whole numbers, and their decimals are not periodic.

Because the number of decimal places for real numbers is infinite, they subtly introduce logical difficulties greater than we can imagine, and these difficulties are shared by the omega numbers. Let us consider them.

First, the infinite number of decimal places means that real numbers are not countable: there is no numbering system $r_0, r_1, \ldots, r_n, \ldots$ that constitutes the complete list of real numbers (we have Cantor to thank for this proof). Thus, the set of real numbers constitutes an infinite set larger than the infinite sets of whole numbers and rational numbers (since, thanks again to Cantor, this set of rational numbers is the same size as the set of whole numbers).

A sometimes neglected consequence of this uncountability is that we cannot compute all the real numbers: computable real numbers are by definition those for which there is a computer program that, allowed to run indefinitely, produces a string of the digits for the number one after the other. But the number of programs is denumerable: we can, for example, count all of them if we consider the sets of programs that have $1, 2, 3, \ldots n, n+1, \ldots$ symbols. Each set contains a finite number of programs, which makes the set finitely countable and, accordingly, the entire set of all programs countable. We will use this numbering of programs later. It follows that there are not enough programs to compute all the real numbers. Of course, every rational number is computable, as are the familiar constants of classical mathematics: $\pi, e \log 2, e + \pi, \sin(1)$, and so on. In each case, their definition (for example, the series $e = 1 + 1/1! + 1/2! + 1/3! + \ldots$) dictates the

program for calculating their digits one by one.

Do we really have to take uncomputable real numbers seriously? Can't we just ignore them? Actually, no, we can't, because computability theory, developed in the 1930s by Kurt Gödel, Alan Turing, and several other logicians, not only demonstrated such numbers theoretically (building on arguments of uncountability). The theory also defined uncomputable numbers with such exquisite precision that they now enjoy a mathematical status comparable to π and e.

A simple idea for defining an uncomputable number will bring us close to omega numbers. It is based on the so-called halting problem. The question of whether a program halts has both theoretical and practical importance: we have all written programs that go into infinite loops. Take, for example, the program

```
c: = 1;
while c > 0 do c: = c + 1;
end
```

A program has only one option: once launched, it can stop after a finite time; or it can run forever.

Let us draw up the list of all the possible programs $P_0, P_1, \ldots, P_n, \ldots$ written, for example, in Java (a ubiquitous programming language) and classify them by size as described above. Now let us consider the real number whose decimal expansion is $\tau = 0.a_0a_1 \ldots a_n, \ldots$ where each a_n equals 1 if the program P_n stops, and 0 if it continues indefinitely.

The undecidability of the halting problem ("it is impossible to write a program A that, examining any other program—here program P_n—returns in finite time whether P_n halts or whether it runs indefinitely"), demonstrated in 1936 by Turing, determined that the real number τ is not computable.

Thus certain numbers, such as τ, are not computable, but they are known, for they can be defined without the least ambiguity. Yet they are unknowable, because no program can produce their string of digits. That is the way the mathematical world works: some of its numbers can be seen (defined), but not touched (computed).

20.2. Omega numbers are much worse

Omega numbers are like τ, but worse. Note that there are many programs (an enumerable infinity) that we know either will halt, for example, the programs `PRINT 0, PRINT 1, PRINT 2, PRINT 3`, and so on, or won't halt. Thus we can know an infinity of digits of the number τ (even if it is still not computable in the general sense). On the other hand, for Chaitin's omega numbers, we can only know a finite number of digits.

What do we mean by "know"? In mathematics, ever since logic formalized strong theories at the beginning of the 20th century, mathematicians have adopted the habit of suggesting (at least implicitly) which theory governs their thinking, and in which formalized language they may write the detailed version of the proofs they propose. ZFC set theory (that is, Zermelo-Fraenkel, with the axiom of choice) is a satisfactory theory for practically all mathematicians. It also serves as a basis for Nicolas Bourbaki's comprehensive treatise "Elements of Mathematics." The proofs that we mention here are proofs that can be formulated in ZFC. We need only remember that when we say "we can know P" or "we prove P," it means that there exists a proof in ZFC that demonstrates P. This remark having been made, we won't repeat it. When we assert that such a property can be demonstrated, we mean "using ZFC axioms."

Chaitin's omega numbers are well-defined real numbers that, as we will see below, are not only not computable (no program can produce their digits one by one, as we have already seen for τ), but we can know only a finite number of their digits. Any mathematical theory is incapable of calculating the digits of the omega numbers, which it nonetheless defines perfectly!

If Ω is a given Chaitin omega number, and if n is a specified integer, one of the following statements is true:

- the nth bit of Ω is a 0,
- the nth bit of Ω is a 1.

Yet as soon as n becomes sufficiently large, neither of these two statements can be proven. Briefly stated, if Ω is a Chaitin number, the Ω digits, except for a finite number of them, are undecidable.

20.3. Pure extract of undecidability

Mathematicians of the 1970s were astonished to realize how unknowable defined objects could be. Yet this quintessence of undecidability, represented by Chaitin's omega numbers, has just been surpassed!

Indeed, while studying Chaitin's omega numbers and using an old theorem demonstrated by Stephen Kleene in 1952 (the recursion theorem), the mathematician Robert Solovay discovered that several of them (which we will call Solovay omega numbers), though well defined, were completely unknowable. In other words, if Ω is a Solovay omega number, none of its digits can be known. Note that Solovay had already acquired a certain celebrity in 1970 for having solved an important problem of logic. Thirty years later, he was the hero of a new feat that contradicts the stubborn idea that a mathematician's productivity declines rapidly with age.

This essential undecidability posed by the Solovay omega numbers shows how the apparently innocuous introduction of numbers with an infinity of decimal places can—once all their consequences are considered—lead to situations bordering on the absurd. At the very least, they can plunge any sensible person into an abyss of perplexity: How can something that is so well defined be perfectly and absolutely unknowable?

20.4. Is there a practical definition of omega numbers?

Chaitin's omega number is "the halting probability of a self-delimiting universal machine." Argh! Let's unpack this definition.

A universal machine U is a machine that is capable of calculating any function defined by a valid program. All modern computers are universal machines; the concept was introduced by Turing in 1936. The requirement that the programs be self-delimiting means that the beginning of any valid program for U cannot itself be a valid program for U. One way to ensure this property is to equip the programming language of the machine with a string indicating the end. For example, we agree to end all U programs with the string of four characters "E" "N" "D" ".", which can only be used once in the same program. DNA contains analogous sequences of bases that indicate the end of a gene.

The fact that programs are self-delimiting (something within them signals their termination) makes it possible to assign a probability to each program P of machine U. To do that, we systematically construct a pro-

A hierarchy of incalculabilities

René Daumal (a French poet, 1908–1944) imagined Mount Analog, a mysterious mountain symbolizing research, whose summit is by definition unreachable despite an accessible base. The various numbers evoked in this article are all comparable to the summit of Mount Analog. Rational numbers are computable and periodic, but their decimal places are infinite.

Transcendantal numbers like π and e are computable: we can never know all their digits, but the difference between these numbers and their approximations is as small as one could wish.

The number τ, whose bits are 0 when the program associated with its nth digit halts, and 1 when it doesn't. We know how to compute an infinite number of these bits, but an infinity of others are unknowable.

The Chaitin Ω numbers, which indicate the probability that a program running on a universal machine will come to a halt: we know how to compute only a finite number of their digits.

The Solovay numbers, for which we cannot compute a single digit, although the numbers are well defined.

Figure 1.

gram P until reaching a string that corresponds to the binary transcription of "E" "N" "D" ".". A program P will be a sequence of n bits such as 0110101101001. The probability of obtaining such a string, thus program P, is 2^{-n}, for each bit has a probability of 1/2 of being the right one. We really are speaking of probability, for one can actually choose programs at random by a process that gives P with the probability 2^{-n} (see Figure 2).

Now imagine that you use this process to randomly generate programs for the universal machine U, and that each time you find a valid program for U, you activate it. Either the program runs forever, or it eventually halts. The sequence of "coin tosses" thus sometimes leads to the program stopping and sometimes to an infinite loop (either because you never find any program that will run or because the programs that run don't stop). On implementing this well-defined process, the probability of U coming to a halt is Chaitin's omega number for the universal machine U, which we

A mandala

The symbol Ω is repeated four times in the structure of this graphic symbol of the universe, called a mandala. The symbol appears in many other mandalas, and represents a fitting artistic rendition of the Ω numbers.

Figure 2.

denote Ω_U. For each universal machine, the number Ω_U is well defined. Moreover, it is as well characterized as the numbers π or e.

For each self-delimiting universal machine U there corresponds a Chaitin omega number; and because there is a countable infinity of such machines U, there is a countable infinity of Chaitin omega numbers Ω_U.

The Solovay omega numbers are defined based on a particular class of universal machines specially concocted so that they get round ZFC theory. The technique for defining Solovay omega numbers consists in transforming a universal machine Ω_U by modifying the initial digits so that ZFC cannot predict a single bit for such a number. This definition is a touch magical, but the details are technically too complicated to present here. That is easily understandable: they have eluded mathematical solution for more than 20 years.

Note that Chaitin's number Ω_U is, by definition, a sum of numbers each of which is equal to 2^{-n}. Specifically, $\Omega_U = \sum 2^{-n}$, the sum being carried out over all n that are the length of U programs that come to a halt.

This definition indicates a practical procedure for approximating Ω_U. You take a certain number of programs (for example, all those whose length i is less than n), let them run for a certain time (for example, through n computing steps), and add 2^{-i} for all the programs that stopped. The increasing sequence x_n thus defined converges to Ω_U.

20.5. Surely you are joking?

You are feeling a little uneasy, and you might even suspect that I am pulling your leg. On the one hand, I claim that Ω_U numbers are not computable (and in the case of the Solovay numbers, even totally unknowable). At the same time, I propose a foolproof method for approximating these Ω_U numbers, in other words, for computing them!

Rest assured: I am not putting you on. Indeed, in this apparent contradiction lies all the subtlety of the omega numbers. If U is a self-delimiting universal machine, we really can construct an increasing sequence of rational numbers that converges to Ω_U, but the convergence happens very slowly—so slowly that you will never be certain of obtaining more than a few precise Ω_U bits. In the case of Solovay numbers, you won't reach even a single digit with any certainty. The convergence to Ω_U is slower than the convergence of any other computable increasing sequence to a computable number (someone recently demonstrated the elegant finding that this exaggerated slowness is a characteristic feature of omega numbers).

For ordinary mathematical constants like π, several computing methods are generally known, each producing a sequence of numbers x_n that converges to the constant. Some techniques are fast (for example, you can

Defining omega numbers

A Chaitin omega number is the halting probability of a self-delimiting universal machine.

A universal machine U is one capable of computing any function that is computable by a program. The programs (assumed to be written in binary) are called self-delimiting if the instruction to stop is contained within the program itself. For example, the string 1111 indicates that the end of the program has been reached.

Testing for success or failure

- Randomly select 0's and 1's using a random procedure, for example, by tossing a coin.
- Continue until you have a program for U. In the case that you do not get there (because you never obtain the string 1111), the result is a failure.
- In the case where you do obtain 1111, you now have a program for U that you submit to U for execution. If U halts while running the program, that constitutes success; if not, consider the result a failure.

Completing the test results either in success or failure. The probability of succeeding is, by definition, the omega number for U. To approximate Ω_U, repeat the test k times, adding up the number of successes m. Then m/k is the approximation.

This definition is satisfying only in theory, for it presupposes that one can continue indefinitely generating 0's and 1's and that it is possible to know that a program will never stop (which constitutes an undecidable problem).

To actually approximate Ω_U, you can make the test more realistic by doing the following. Pick a number n. Generate a set of 0's and 1's, stopping only when you have obtained 1111 or when you have generated n digits. If you reach n digits without obtaining 1111, the test has failed. To determine whether a program has stopped, run it for n seconds. If it has not stopped at the end of these n seconds, the test has failed. By repeating the test (this time totally realistic) k times, and by considering the number of successes m over the number of tries k, you obtain an approximation of Ω_U.

Figure 3.

obtain several new digits in going from x_n to x_{n+1}. Other approaches can be slower. Number crunchers of course prefer more rapid techniques; and you could say that, faced with mathematical constants, their skill consists in inventing fast methods of convergence. When you are dealing with a Chaitin omega, you know definitely and absolutely that this approach is useless. Not only will no method be fast, but no means of approximation will enable you to know how rapidly it can supply bits for the omega constant computed.

20.6. The properties of omega numbers

It is possible to imagine all the universal machines and all the omega numbers associated with them. The infinite class of Chaitin omega numbers is thus countable, as is that of Solovay omega numbers. Moreover, we know a lot about them. There is no paradox in the fact that it is possible to demonstrate specific properties of the omega numbers (including the Solovay omega numbers), even though their bits are not computable. In the real world, to have access to general knowledge – for example, "the average weight of Americans is greater than the average weight of Europeans"—you have to gather detailed information. In the mathematical world, that is not always the case: it is possible to know something general about a number Ω_U—for example, "the frequency of 1's and 0's is the same in Ω_U binary notation"—and at the same time not know a single specific Ω_U bit. Yet another mathematical enigma!

Here are some of the known properties of omega numbers:

- All omega numbers are irrational and transcendantal (no polynomial equation containing whole-number coefficients has an omega number as its solution).
- The decimals of all omega numbers are uniformly distributed: the set of their digits in base 10 carries a tenth of "0," a tenth of "1," ..., a tenth of "9," and there is an analogous property in every other numbering system.
- Each omega number is a "universal number" in each base: every finite sequence of digits is present in it. One could even say that each omega number contains every finite sequence of n decimal digits with a frequency of 10^{-n} (of course, there is an analogous property in all numbering systems). Consequently, for all omega numbers, we know that somewhere there is a series of a billion consecutive 0's (nothing

like that has been demonstrated for constants such as π and e).

- All omega numbers are random in the strictest mathematical sense (the technical term is "Martin-Löf random" in honor of the Swedish mathematician who introduced this concept in 1966). That implies, in particular, that (a) a program for predicting the nth bit of an omega number based on $n-1$ initial bits can never be better than chance; (b) if we extract a subsequence of the sequence of digits of some omega numbers using an algorithm (for example, by retaining only the digits whose position number is a prime), this sequence will be that of digits of an irrational, transcendental, uniformly distributed, random number, and in fact even another Chaitin omega number.
- All omega numbers are uncompressible. Specifically, for each omega number Ω_U, there is a constant, c, such that the shortest program giving the n first few bits of Ω_U is at least as long as $n-c$ (it is not possible save more than c bits of information when attempting to compress an initial Ω_U string).
- All omega numbers are uncomputable, and yet each is the limit of a computable increasing sequence of rational numbers (they are said to be approximable; see Figure 4). This convergence is slower than the convergence for any computable sequence of rational numbers to a computable number.
- An omega number can begin with any finite string of digits. Thus there is an omega number that begins with 3, 14, another that begins with 3, 1415, another that begins with 3, 141592, and so on. Note, however, that universal machines that have those particular numbers for omega numbers will be artificially constructed.
- If the sum of two omega numbers is less than 1, the sum is an omega number; likewise for the product (these elegant properties are not true for irrational numbers or for transcendental numbers [for example, $\pi/4 + (2 - \pi/4) = 2$]).

The fundamental principle common to all these properties is the fact that knowing the first few digits n of the Chaitin number Ω_U associated with the universal machine U makes it possible to know whether all U programs shorter than or equal to n stop (whereas it would take 2^n bits of the number τ for U to know the same thing). In other words, Ω_U contains a superconcentrated form of information about the undecidable halting problem of U programs. In theory, many conjectures could be resolved if we knew the first 10,000 bits of a Ω_U for a "natural" universal machine U (for

example, that associated with the Java language that runs your computer).

A conjecture such as "Every even number greater than 2 can be written as the sum of two primes" (Goldbach's conjecture) is essentially equivalent to a program searching endlessly for a counterexample, a program that is only several hundred bits long. All conjectures of the form "ZFC enables proof of P" if P is a fairly short statement could also be resolved (theoretically) by knowing a few hundred omega bits of a natural universal machine.

Thus omega numbers not only distill information about halting programs. They are also concentrates of mathematical information.

To console us for the fact that we will never know even 1,000 bits of a "natural" omega number, we can tell ourselves that extracting information from omega numbers is a finite but incredibly long job (hence my use of "theoretically" in the preceding paragraph). Consequently, even if you know 1,000 bits of a natural Chaitin omega number, you could never really use it. As Martin Gardner and Charles Bennett have written: "Omega is in many senses a cabalistic number. It can be known of through human reason, but not known. To know it in detail, one must accept its uncomputable sequence of digits on faith, like words of a sacred text."

References

[1] C. H. Bennett and M. Gardner, The random number omega bids fair to hold the mysteries of the universe, *Scientific American*, vol. 241, pp. 20–34, 1979.

[2] C. S. Calude, Chaitin Ω numbers, Solovay machines, and incompleteness, *Theoretical Computer Science*, vol. 284, pp. 269–277, 2002. See `http://citeseer.ist.psu.com/calude99chaitin.html` for a coherent and exhaustive treatment.

[3] J. P. Delahaye, *Information, complexité et hazard*, 2d edn., Hermes Science Publication, Paris, 1999.

[4] J. P. Delahaye, *L'Intelligence et le calcul*, Pour la science, Belin, Paris, 2002.

[5] R. M. Solovay, A version of Ω for which ZFC cannot predict a single bit, in C. S. Calude, G. Păun, *Finite Versus Infinite: Contribution to an Eternal Dilemma*, pp. 323–334, Springer, London, 2000.

Computable and approximable numbers

A. By definition, a real number x (between 0 and 1) is computable if it is the limit of an increasing computable sequence of rational numbers $x_n = p_n/q_n$. Furthermore, this sequence must satisfy that for every integer $n : |x_n - x| < 2^{-n}$.

Classical constants are all computable real numbers, since for each x, a sequence of rational numbers that converges to x is known, and it is possible to bound the errors (for example, to stipulate that the error made by x_n is less than 2^{-n}).

The definition can be shown to be equivalent to:

- x can be written in binary in the form: $x = 0.a_0a_1a_2\ldots$ with a function $n \to a_n$ (each a_i equals 0 or 1) that is computable by a program (in other words, there is a program that produces the bits of x);
- there is a program that produces all the rational numbers less than x, and another that produces all the rational numbers greater than x.

B. A real number is approximable (computably enumerable, in mathematical jargon) if, by definition, it is the limit of an increasing computable sequence of rational numbers $x_n = p_n/q_n$ (as for computing, but omitting the second condition bearing on the importance of the error). An equivalent definition is the following: there is a program that produces the rational numbers less than the real number x in question.

The difference between approximable numbers and computable numbers appears subtle. But in fact it is enormous, and the Chaitin omega numbers are precisely numbers that are simultaneously approximable and uncomputable. The approximable numbers are all well defined (we know sequences that converge to them), and yet some of them, like the omega numbers, are not computable. Moreover, Solovay showed that certain omega numbers have the characteristic that we cannot know any of their digits with certainty. They are well defined, yet absolutely unknowable.

Figure 4.

Calculating some omega digits

There are processes based on self-delimiting universal machines that enable construction of other machines (so-called artificial machines) that are also universal. But the omega numbers of these machines begin with a few binary numbers determined in advance, for example, 01010101010101.

The omega numbers associated with this type of artificial universal machine are themselves artificial, and their initial digits (chosen arbitrarily) carry no information.

In contrast, knowing the digits of a particular self-delimiting universal machine U that has not been constructed with digits known in advance is a very difficult task. For Ω_U contains compressed information about the termination of U programs. Is it possible to compute a handful of bits for such an Ω_U?

Yes! and that is precisely what Cristian Calude, Michael Dinneen, and Chi-Kou Shu recently did. First they defined the most natural universal machine possible by considering the simplest instruction set possible and by adopting the most concise notation they could. Next, they tried to predict the behavior of a large number of short U programs. They succeeded in analyzing all the programs whose length was shorter than 84 bits (some programs halted; others did not). They were able to deduce with certitude the first 64 bits of Ω_U. These bits are:

0000001000000100000110001000011010001111110010111011101000010000

Their achievement represents the first attempt to precisely compute the digits of a random number. Note that this is only a game, for in reality, 64 bits is hardly sufficient even to think of tackling conjectures of any interest. One could try going beyond 64, but stumbling blocks arise quite rapidly, and it is unlikely that such calculations could help in resolving open mathematical problems.

Figure 5.

Ω holds the secret of all mathematical enigmas

If it were possible to know the omega number Ω_U of a universal machine, for example, through transcendental meditation (or even if one succeeded in discovering the first 10,000 bits), one could resolve the essence of all the questions that puzzle mathematicians. Let me justify this assertion in two steps:

A. Knowing m bits of Ω_U makes it possible to know whether any program P of U that is shorter than m halts or not, using the following procedure.

Calculate the sequence of terms of an increasing sequence x_n that converges to Ω_U. To determine x_n, take all the programs whose length i is shorter than n and run them for n computing steps. Next, sum the probabilities 2^{-i} of each of the programs that have halted, which gives x_n. At some point, the number x_n written in base 2 will have the same m initial bits as Ω_U. We assume we know these m bits, so we will be able to recognize when it happens. Now, at this stage in the computation, either P has been computed in x_n, for the program has stopped (thus we know that P is a program that terminates). Or it has not stopped, and we know that it will never stop, for its probability 2^{-m} added to x_n will exceed Ω_U.

B. All the open conjectures of mathematics can be expressed as follows: "Can the formal system ZFC prove P?" For P, we take the statement expressing the conjecture. Each of these questions is equivalent to a U halting problem, specifically, halting a U program that enumerates valid proofs of ZFC until it finds a proof of P. Thus, if we know a number of Ω_U digits greater than the length of the computing program that corresponds to P, we have a way of responding to the question, "Can the formal system ZFC prove P?" Once implemented, this method will return a response in finite time. Unfortunately, the computing time threatens to be very long and, in any event, impossible to predict in advance.

Figure 6.

Chapter 21

Chaitin and Civilization 2.0

Tor Nørretranders

Strandvejen 413, DK-2930 Klampenborg, Denmark; tor@tor.dk

A short story on why stories have to be long: Why Chaitin did not meet Gödel or Leibniz or Plato during his random walk of life. Why he is to become the hero of a new age, The Link Age where everything is being connected to everything else.

21.1. Chaitin on the beach

"It snowed!" A sense of desperation darkens the otherwise childishly bright, shining and smiling eyes beaming from Gregory Chaitin's face. The red cap on his head protects him from the burning hot August sunshine flooding the beautiful long, sandy beaches of Fire Island, a few hours drive from Manhattan where Chaitin picked me up at my hotel in the morning. It was our first meeting, fifteen years ago. The despair is evident as Chaitin answers my question on whether he ever met Kurt Gödel, the German mathematician who, in 1931, famously proved that one cannot prove everything. There are things we humans can know, but not prove, Gödel showed to the world.

Gödel had been the hero of the young Chaitin. Already in college, in the sixties, Chaitin had started showing that the results of Gödel were not, in fact, a mathematical curiosity with limited scope. It was a basic fact about the world (or at least our description of it): No formal description can ever grasp everything, there will always be aspects of reality escaping our description. A complete description cannot be consistent and a consistent description cannot be complete. It is exactly like life as we know it, everyday life right there as it is, but Gödel showed that it is also the case for the purest of the pure mathematical problems.

Life is full of contradictions, confusion and sticky ends. We all know

that. We might have hoped for purity and majestic harmony in math and pure science. But no! Gödel has shown us that this is not the way it is, at least in some extremely elegantly thought out cases that he studied back in 1931. Greg Chaitin has shown us that this result is not a weird detail, but the way things are—or at least the way our descriptions are. It is the same wisdom that we find in quantum physics: We can never give a description that is at the same time complete and without contradiction. Danish physicist Niels Bohr coined the phrase complementarity to describe the fact that when it comes to electrons and other sub-atomic entities, we have to use more than one concept to catch all of their behaviour (like both the concept of waves and particles). Yet at the same time we have to accept that these two concepts contradict each other. We need them both, but they exclude each other.

Quantum mechanics and Gödel's proof is certainly not the same thing, but they are more related than we usually see them. Epistemologically, they tell the same story: Any description is incomplete, we need more than one description to grasp anything, but these different descriptions will never be united, because they contradict each other.

And here was a guy with a red baseball cap walking a mile-long sandy beach in the summer heat telling me about snow. "Snow is fantastic! Cross-country skiing is just like a mathematical abstraction! There is only the blue sky and the white, snow-covered mountain", explained Chaitin, telling me how a common friend has taught him to ski. The snow-covered land is like pure math: The contours and general outline of the landscape is there, but all the messy details like bushes and small trenches and streams are evened out by the deep snow-cover. Snow abstracts away the landscape. Chaitin likes that. But he didn't like the snow 35 years ago. It meant that he never met Gödel, even though he did have an appointment.

"His secretary called and said that Gödel was worried about his health and wouldn't go out into the snow." Chaitin was to take the train from New York to Princeton to meet the master. But there was snow. Soon after he had to leave for Buenos Aires where his parents worked for the UN and he worked for IBM.

That was the shortest answer, Chaitin could give to the question, if he ever met Gödel. The was no simpler structure to the story, it was a chain of coincidences, not structured in a simpler way.

21.2. Random walk

The snow-story on the beach is, of course, a miniature of Chaitin's first major result: That the elusive phenomenon of randomness is all about how simple something is to describe. A structured and ordered thing is simple and can be described in a short way. That is what we mean by order. Something random, on the other hand, cannot be described in any simple way. That is what we mean by randomness. In fact, if something is totally random and has no structure or order at all, it is its own shortest description. One cannot say it any simpler than it does itself. You have to tell all the details to get anywhere. You have to tell about the snow.

Walking along the beaches of Fire Island was a surprise to a European. Such beautiful beaches this close to the teeming megapolis at Manhattan. And yet not a single sunbathing or swimming human being. Not one! After maybe an hour of walking and talking about snow and the like, I asked Chaitin: "Where is everyone?"

A few miles more of walking, and there they were, everyone. Hundreds and hundreds of people crammed in a very small area on the beach. Lying side by side and diving into the ocean waves, almost hand in hand. Crammed! Why this, why miles and miles of just sand with a few crab shells, and then, suddenly, hundreds and hundreds of people packed into a few hundred meters of coast line?

"Insurance," Chaitin explained. There had to be a lifeguard wherever people dipped into the ocean. So people were ordered to stay close to a lifeguard post. Using the rest of the beach was simply not allowed.

Central surveillance and control doesn't allow for a random scattering of people at a beach. They have to be concentrated and packed so that they can be saved more easily.

Simplicity is control. Randomness is messy and difficult to control.

21.3. Worldview

Let's be honest for a moment. We don't know a lot about the world. But we know a lot about our description of the world. Whether the description tells us a lot about the world, we don't really know. Science is not about the world, but about what we can say about the world, physicist Niels Bohr said. About the world in which we speak.

Hence, there is room for quite a few models. I want to explore the simplest possible model of the world: The world is random. It is fully linked.

Everything is linked together in such a way that everything is connected to everything else. Nothing can be described unless everything else is described, because there are so many causal connections that you cannot omit anything without leaving out something that could be of importance. Chaos theory tells us that we can never allow ourselves to ignore even fine differences in initial conditions. Quantum mechanics tells us that everything is entangled with everything else.

Yet, we see a world of things that are separate and often can be described locally, that is without reference to the rest of the universe. How come there exists a kind of local structure that is decoupled from the web of everything being connected to everything? My table is stable, no matter what goes on on Saturn.

How come that in a world of connections, a world of links, there is local stability, things we can describe without describing everything else? I like to think of it as a kind of boiling. In hot water, approaching the boiling point, small pockets of vapour will form where ever the vapour pressure of the steam is higher than the pressure from the water around it. Small pockets of non-water arise in the water. Likewise, in a world of infinitely many connections between all the ingredients, local bubbles of stability arise.

When a macroscopic object is formed, it will be big enough that all the odd quantum entanglements just vanish in importance. It may interact with the outside world, but the interactions are so tiny and all equal out. Therefore we can describe the chair with out referring to the weather on Jupiter, or at least we think so—for any practical purpose. So everything is really very random, but sometimes order bubbles out of the pot.

21.4. Leibniz

But why didn't Chaitin meet Leibniz? Well, for starters Leibniz died 291 years ago. Another reason could be that Leibniz was a loner, and Chaitin is another one.

But Leibniz had the very simple idea that whenever one could reduce data to a simple theory and re-deduce the data from the simple theory, it was a good (or even true) theory. This, of course, is very much like Chaitin's ideas.

So: The world is random and you cannot describe anything without describing everything. But locally there is structure, simplicity in the mess of links and connections, like bubbles in boiling water. Sometimes, we can

take the simple structures and use them again and again, according to the laws for their behaviour, and we can get the original random mess back! In that case we feel we understand: We can take mess, reduce it to simplicity, do some push-ups with the simplicity and get the mess back.

Let's translate that into randomness and order: We take randomness, find some structure in it, manipulate the structure and get randomness back.

Let's translate it into science: We have reality (what a mess!) and we extract simple principles, manipulate them and get the mess back.

Let's translate it into epistemology: We have a world with no structure, we build structures in our head, and we can reconstruct the mess.

Chaitin meets Leibniz.

21.5. Abstractions?

Plato, the old Greek philosopher, insisted that there were ideas behind the phenomena: All existing horses were the incarnation of the principle of the horse. Ideas before phenomena. That is, order before randomness.

But Plato was wrong. The world is random and there are only approximative concepts.

Or is it so? Most mathematicians (but not Chaitin) are closet-platonists. They think mathematical objects exist before that messy and low world in which we exist.

I am arguing here that the mess is the starting point, order evolved. Some would argue that order was designed and came first.

But perhaps it was in fact designed: 10,000 years ago humans started doing agriculture. Before that we lived a rich and long life as hunter-gatherers, collecting a huge variety and self-grown, wild plants and fruits and hunting down self-grown animals and fish.

But then, after a climate disaster at the end of the Last Ice Age, ocean water level rose by 100 meters (!) and something new had to happen. Agriculture became the answer.

Agriculture is dull: No longer was there an almost infinite variety of plants and animals (with hundreds of species being caught or collected every day). It was all reduced to a few, high-yielding plants and domesticated animals.

Look at the field and say "Wheat!" You don't really have to say more. Look at the wilderness and it will take you a long time to describe.

Agriculture introduces the abstraction. The idea of wheat is in fact

primary to the wheat field. First was the idea, then came the reality. Idea before phenomenon. Agriculture.

21.6. The Link Age

But now water levels are rising again and we have to rethink civilisation, we have to invent a Civilisation 2.0, as I call it. We are entering what I call The Link Age, the era of network links and everything being linked to everything else. We are also leaving the starch-producing agricultural era of a few grasses grown in huge quantities (grasses like wheat, barley, rye, rice, corn, sugar cane, etc.) 10,000 years ago, a rich variety of different individual plants in the self-grown wilderness of the hunter-gatherer culture was transformed into a new reality of stereotypes of cultivated land.

Now, however, Civilisation 2.0 is on its way. It is being created these days, by mostly unknowing wind mill farmers and Web 2.0 collectives of co-operative software producers. The wilderness, the distributed control and the sunshine is coming back.

Agriculture created Civilisation 1.0 with all its real abstractions like the wheat field or the chickens in the barn.

We are now seeing the advent of a Civilisation 2.0 based on the link as the "atom": Networks, peer-to-peer on the net, social software, social technology, keeping tracks of your relationships digitally; renewable energy, recycling, environment. The age of the link, as opposed to the age of atoms and substance. Not things but relationships between things; not individual people, but links between people. The Link Age.

Not hierarchies, not central control, not life guards, but Web 2.0 - spontaneous collaboration. High information content, mediated by machines. Green and hi tech at the same time.

Civilisation 2.0 is arising these days. The world is slowly understanding the severity of the climate crisis and the fantastic opportunities offered by the internet and new epistemology. Randomness will reign again.

Civilisation 2.0 will have theoretical heroes. Greg Chaitin will be the leading one. He discovered the nature of randomness, non-order, non-control, the world as it is in all its richness. Chaitin transcended the logic of the agricultural mind—without even trying. He just did it. He didn't know he did it. He didn't even try. We should all be thankful. You could not give a shorter version of the story than telling about all that Chaitin did. There was no plan. His career has not been centrally organised and planned. It was a crooked, zigzag way, information rich and random. It

was what it was. Never close to the life guard. High risk. High importance.

Greg Chaitin's time is just about to come.

Congratulations, everyone.

Chapter 22

Some Modern Perspectives on the Quest for Ultimate Knowledge

Stephen Wolfram

Wolfram Research

Dedicated to Gregory Chaitin on the occasion of his sixtieth birthday, these remarks attempt to capture some of the kinds of topics we have discussed over the course of many enjoyable hours and days during the past twenty-five years.

The spectacular growth of human knowledge is perhaps the single greatest achievement in the history of civilization. But will we ever know everything? Three centuries ago, Gottfried Leibniz had a plan. In the tradition of Aristotle, but informed by two more millennia of mathematical development, he wanted to collect and codify all human knowledge, then formalize it so everything one could ever want to know could be derived by essentially mathematical means. He even imagined that by forming all possible combinations of statements, one could systematically generate all possible knowledge. So what happened to this plan? And will it, or anything like it, ever be achieved?

Two major things went wrong with Leibniz's original idea. The first was that human knowledge turned out to be a lot more difficult to formalize than he expected. And the second was that it became clear that reducing things to mathematics wasn't enough. Gödel's Theorem, in particular, got in the way, and showed that even if one could formulate something in terms of mathematics, there might still be no procedure for figuring out whether it was true.

Of course, some things have gone better than Leibniz might have imagined. A notable example is that it's become clear that all forms of information—not just words—can be encoded in a uniform digital way.

And—we think—all processes, either constructed or occurring in nature, can be encoded as computations.

Science has gone OK since Leibniz's time, but in some ways not great. Descartes had thought that within a hundred years of his time there'd be a complete theory of our universe, from which everything we might want to know could be calculated. And in some areas—especially the traditional physical sciences—there's been excellent progress. And we've been able to achieve immense amounts in engineering and technology on the basis of that progress. But in other areas—notably the biological and social sciences—there is still rather little that we can calculate. And even in physics, we of course don't have an ultimate theory of our universe.

What about mathematics? In some ways it's hard to assess progress. But I think we'd have to say that it's been mixed. Some of the widely discussed mathematical problems of Leibniz's day have firmly been solved. But plenty have not. Just like the Pythagoreans, we still don't know whether a perfect number can be odd, for example.

So what happened with science and mathematics? Why did they turn out to be difficult? Did we just not have enough clever ideas? Or put enough effort into them? I don't think so. I think there's a fundamental problem—a fundamental barrier to knowledge.

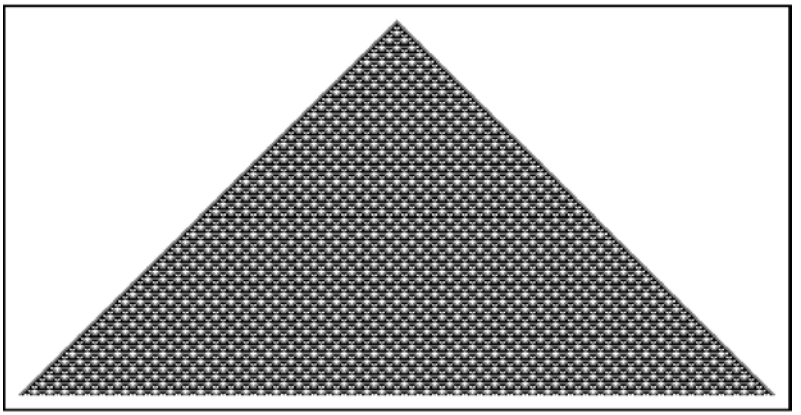
One can think about it as all being related to what I call computational irreducibility. And it depends critically on what is probably the greatest intellectual advance of the past century: the notion of universal computation.

Here's the point of computational irreducibility. Imagine a system that evolves in a certain way. Now ask the question: can we work out what the system will do by spending less computational effort than it takes the system itself? Can we find a shortcut that will determine what will happen in the system without having to follow all the steps that the system itself goes through?

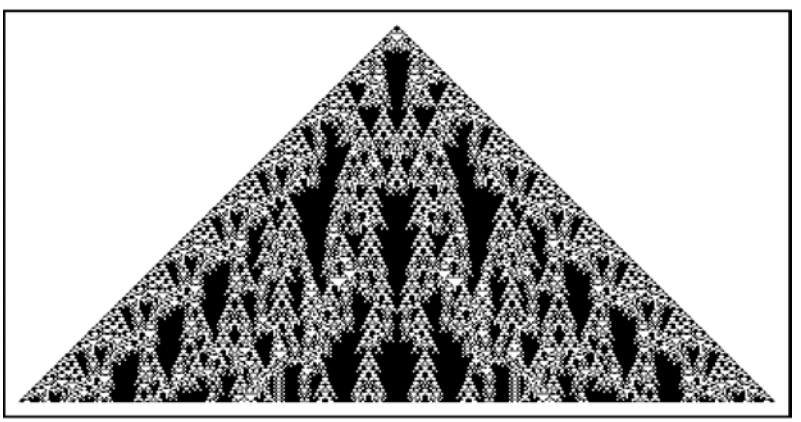
The great triumphs of the traditional exact sciences have essentially all been based on being able to do this. To work out where an idealized Earth orbiting an idealized Sun will be a million years from now, we don't have to trace the Earth around a million orbits: we just have to fill a number into a formula and immediately get a result.

But the question is: will this kind of approach always work? Look at

the second picture below.



Picture 1



Picture 2

Is there a way to shortcut what is happening here, to find the outcome without explicitly following each step? In the first picture above, it's obvious that there is. But in the second picture, I don't think there is. I think what is happening here is a fundamentally computationally irreducible process.

If one traces each step explicitly, there is no problem working out what will happen. But the point is that there is no general shortcut: no way to find the outcome without doing essentially as much work as the system itself.

How can this be? We might have thought that as our methods of mathematics and science got better, we would always be able to do progressively better. But in a sense what that would mean is that we, as computational systems, must always be able to become progressively more powerful. And this is where universal computation comes in. Because what it shows is that there is an upper limit to computational capability: once one is a universal computer, one can't go any further. Because as a universal computer, one can already emulate anything any other system can do. So if the system one's looking at is a universal computer, it's inevitable that one can't find a shortcut for what it does.

But the question for science—and for knowledge in general—is how often the systems one's looking at are universal, and really behave in computationally sophisticated ways.

The traditional successes of the exact sciences are about cases where the systems one's looking at are computationally quite simple. And that's precisely why traditional science has been able to do what it does with them. They're computationally reducible—and so what the science has done is to find reductions. Find things like exact formulas that give the outcome without working through the steps.

But the reason I think science hasn't been able to make more progress is precisely because there are lots of systems that aren't computationally reducible. There are lots of systems that can—and do—perform sophisticated computations. And that are universal. And that are just as computationally sophisticated as any of the methods we're able to use to analyze them. So that they inevitably seem to us "complex"—and we can't work out what they will do except with an irreducible amount of computational work.

It's a very interesting question of basic science just how ubiquitous computational irreducibility really is. It's a little confusing for us, because we're so used to concentrating on cases that happen to be computationally reducible. Most of our existing engineering is built on systems that happen to behave in computationally reducible ways—so that we can readily work out what they'll do. Biological evolution, as well, tends to have an easier time dealing with computationally reducible systems. And we as humans

inevitably tend to notice those aspects of systems that are computationally reducible—because that is what our powers of perception allow us to recognize.

But it is possible to do what amounts to a more unbiased study. The basic idea is just to look at all possible simple computational systems—say all possible small programs of a certain form. In effect, to do a new kind of empirical science, and to look out into the computational universe, and see what's there. Well, this is something I have spent a great deal of time doing. And one of the big things I've concluded is that in fact computational sophistication—and computational irreducibility—is quite ubiquitous. Indeed, I have formulated what I call the Principle of Computational Equivalence, which in effect says that almost any time one sees behavior that does not look obviously simple, it will turn out to be of equivalent computational sophistication. And what this means is that in a sense almost everywhere outside the places where the exact sciences have already been able to make progress, there will be fundamental limits to progress.

Certainly progress is not impossible. In fact, as a matter of principle, there must always be an infinite hierarchy of pockets of reducibility—an endless frontier for traditional science. But there will also be lots of computational irreducibility. Still, computational irreducibility certainly does not prevent science from being done. It just says that the expectations for what can be achieved should be different.

It puts pressure on having the simplest possible underlying models. Because it says that one has no choice but in effect just to follow every step in their behavior. As a practical matter, though, it's often perfectly possible to do that: to find by simulation what a system will do. And that's something that generates a lot of very useful knowledge. Indeed, it's becoming an increasingly critical element in all sorts of technology and other applications of science.

So what about mathematics? In the abstract, it's not even obvious why mathematics should be hard at all. We've known for a hundred years that the axioms on which all our current mathematics is based are quite simple to state. So we might have thought—as most mathematicians did in the early twentieth century—that it'd just be a question of setting up the appropriate machinery, and then we'd systematically be able to answer any question we might pose in mathematics. But then along came Gödel's Theorem. Which showed that there exist at least some questions that can

be formulated in mathematical terms, but can never be answered from its axioms. But while Gödel's Theorem had a big effect on thinking about the foundations of mathematics, I think it's fair to say that it's had almost no effect on the actual practice of mathematics. And in a sense this isn't surprising. Because the actual question that Gödel talked about in proving his theorem is a weird logic-type thing. It isn't a kind of question that ordinary mathematicians would ever naturally ask.

But the real heart of Gödel's Theorem is the proof that standard mathematical axiom systems are computation universal. And a consequence of this is that mathematics can in effect show computational irreducibility. Which is, at a fundamental level, why it can be hard to do. And also why there can be questions in it that are formally undecidable from its axioms.

But the question is: how common is something like undecidability? Practicing mathematicians always tend to assume it's rare. But is that really true? Or is it just that the mathematics that gets done is mathematics that avoids undecidability? I firmly believe it's the latter.

It's often imagined that mathematics somehow covers all arbitrary abstract systems. But that's simply not true. And this becomes very obvious when one starts investigating the whole computational universe. Just like one can enumerate possible programs, one can also enumerate "possible mathematicses": possible axiom systems that might be used to define mathematics. And if one does that, one finds lots and lots of axiom systems that seem just as rich as anything in our standard mathematics. But they're different. They're alternative mathematicses. Now, in that space of "possible mathematicses" we can find our ordinary mathematics. Logic–Boolean algebra–turns out for example to be about the $50{,}000^{\text{th}}$ "possible mathematics" that we reach. But this kind of "sighting" makes it very clear that what we call mathematics today is not some absolute thing. It's just a particular formal system that arose historically from the arithmetic and geometry of ancient Babylon. And that happens to have grown into one of the great cultural artifacts of our civilization.

And even within our standard mathematics, there is something else that is going on: the questions that get asked in a sense always tend to keep to the region of computational reducibility. Partly it has to do with the way generalization is done in mathematics. The traditional methodology of mathematics puts theorems at the center of things. So when it comes to working out how to broaden mathematics, what tends to be done is to ask

what broader class of things still satisfy some particular favorite theorem. So that's how one goes from integers to real numbers, complex numbers, matrices, quaternions, and so on. But inevitably it's the kind of generalization that still lets theorems be proved. And it's not reaching anything like all the kinds of questions that could be asked—or that one would find just by systematically enumerating possible questions.

One knows that there are lots of famous unsolved problems in mathematics. Particularly in areas like number theory, where it's a bit easier to formulate possible questions. But somehow there's always been optimism that as the centuries go by, more and more of the unsolved problems will triumphantly be solved.

I doubt it. I actually suspect that we're fairly close to the edge of what's possible in mathematics. And that quite close at hand—and already in the current inventory of unsolved problems—are plenty of undecidable questions. Mathematics has tended to be rather like engineering: one constructs things only when one can foresee how they will work. But that doesn't mean that that's everything that's there. And from what I've seen in studying the computational universe, my intuition is that the limits to mathematical knowledge are close at hand—and can successfully be avoided only by carefully limiting the scope of mathematics.

In mathematics there has been a great emphasis on finding broad methods that in effect define whole swaths of computational reducibility. But the point is that that computational reducibility is in many ways the exception, not the rule. So instead, one must investigate mathematics by studying—in more specific terms—what particular systems do.

Sometimes it is argued that one can see the generality of mathematics by the way in which it successfully captures what is needed in natural science. But the only reason for this, I believe, is that natural science has been limited too—in effect to just those kinds of phenomena that can successfully be captured by traditional mathematics!

Sometimes it is also said that, yes, there are many other questions that mathematics could study, but those questions would "not be interesting". But really, what this is saying is just that those questions would not fit into the existing cultural framework of mathematics. And indeed this is precisely why—to use the title of my book—one needs a new kind of science to provide the framework. And to see how the questions relate to questions of undeniable practical interest in natural science and technology.

But OK, one can argue about what might or might not count as mathematics. But in physics, it seems a bit more clear-cut. Physics should be about how our universe works.

So the obvious question is: do we have a fundamental theory? Do we have a theory that tells us exactly how our universe works?

Well, physics has progressed a long way. But we still don't have a fundamental theory. Will we ever have one? I think we will. And perhaps even soon.

For the last little while, it hasn't looked promising. In the nineteenth century, it looked like everything was getting wrapped up, just with mechanics, electromagnetism and gravity. Then there were little cracks. They ended up showing us quantum mechanics. Then quantum field theory. And so on. In fact, at every stage when it looked like everything was wrapped up, there'd be some little problem that ended up not being so little, and inevitably making our theory of physics more complicated.

And that's made people tend to think that there just can't be a simple fundamental theory. That somehow physics is a bottomless pit.

Well, again, from studying the computational universe, my intuition has ended up being rather different. Because I've seen so many cases where simple rules end up generating immensely rich and complex behavior. And that's made me think it's not nearly so implausible that our whole universe could come from a simple rule.

It's a big question, though, just how simple the rule might be. Is it like one or two lines of *Mathematica* code? Or a hundred? Or a thousand? We've got some reason to believe that it's not incredibly complicated—because in a sense then there wouldn't be any order in the universe: every particle would get to use a different part of the rule and do different things. But is it simple enough that, say, we could search for it? I don't know. And I haven't figured out any fundamental basis for knowing. But it's certainly not obvious that our universe isn't quite easy to find out in the computational universe of possible universes. There are lots of technical issues. If there's a simple rule for the universe, it—in a sense—can't have anything familiar already built in. There just "isn't room" in the rule to, say, have a parameter for the number of dimensions of space, or the mass of the electron. Everything has to emerge. And that means the rule has to be about very abstract things. In a sense below space, below time, and so on.

But I've got—I think—some decent ideas about ways to represent those various abstract possible rules for universes. And I've been able to do a little bit of "universe hunting".

But, well, one quickly runs into a fundamental issue. Given a candidate universe, it's often very obvious what it's like. Perhaps it has no notion of time. Or some trivial exponential structure for all of space. Stuff that makes it easy to reject as being not our universe. But then—quite quickly—one runs into candidate universes that do very complicated things. And where it's really hard to tell if they're our universe or not. As a practical matter, what one has to do is in a sense to recapitulate the whole history of physics. To take a universe, and by doing experiments and theory, work out the effective physical laws that govern it. But one has to do this automatically, and quickly. And there's a fundamental problem: computational irreducibility.

It's possible that in my inventory of candidate universe is our very own universe. But we haven't been able to tell. Because going from that underlying rule to the final behavior requires an irreducible amount of computational work.

The only hope is that there are enough pieces of computational reducibility to be able to tell whether what we have actually is our universe. It's a peculiar situation: we could in a sense already have ultimate knowledge about our universe, yet not know it.

One thing that often comes up in physics is the idea that somehow eventually one can't ever know anything with definiteness: there always have to be probabilities involved. Well, usually when one introduces probabilities into a model, it's just a way to represent the fact that there's something missing in the model—something one doesn't know about, and is just going to assume is "random". In quantum theory, probabilities get elevated to something more fundamental, and we're supposed to believe that there can never be definite predictions for what will happen. Somehow that fits with some peoples' beliefs. But I don't think it scientifically has to be true. There are all kinds of technical things—like Bell's inequality violations—that have convinced people that this probabilistic idea is real. But actually there are technical loopholes—that I increasingly think are what's actually going on. And in fact, I think it's likely that there really is just a single, definite, rule for our universe. That in a sense deterministically specifies

how everything in our universe happens. It looks probabilistic because there is a lot of complicated stuff going on that we're not seeing—notably in the very structure and connectivity of space and time. But really it's all completely deterministic. So that in some theoretical sense we could have ultimate knowledge of what happens in the universe.

But there's a serious problem: computational irreducibility. Even though we might know an underlying deterministic rule, we'd have to go through as much computational work as the universe to find out its consequences. So if we're restricted—as we inevitably are—to doing that computational work within the universe, then we can't expect to "outrun" the universe, and derive knowledge any faster than just by watching what the universe actually does. Of course, there are exceptions—patches of computational reducibility. And it's in those patches that essentially all of our current physics lies. But how far can we expect the computational reducibility to go? Could we for example answer questions like: "is warp drive possible?" Some of them, probably yes. But some of them, I expect, will be undecidable. They'll end up—at least in their idealized form—boiling down to asking whether there exists some set of masses that can have such and such a property, that will turn out to be an undecidable question.

Normally when we do natural science, we have to be content with making models that are approximations. And where we have to argue about whether we've managed to capture all the features that are essential for some particular purpose, or not. But when it comes to finding an ultimate model for the universe, we get to do more than that. We get to find a precise, exact, representation of the universe, with no approximations. So that, in a sense, we successfully reduce all of physics to mathematics. So that we would have, in a sense, achieved Leibniz's objective—of turning every question about the world into a question about mathematics. And this would certainly be exciting. But at some level it would be a hollow victory: for even knowing the ultimate rule, we are still confronted with computational irreducibility. So even though in some sense we would have achieved ultimate knowledge, our ability to use it would be fundamentally limited.

Before one knew about computational irreducibility, one might have imagined that knowing the ultimate laws of the universe would somehow immediately give one deterministic knowledge of everything—not only in natural science, but also in human affairs. That somehow knowing the laws of the universe would tell us how humans would act—and give us a way

to compute and predict human behavior. Of course, to many people this always seemed implausible—because we feel that we have some form of free will. And now, with computational irreducibility, we can see how this can still be consistent with deterministic underlying laws. That even if we know these laws, there's still an irreducible distance—an irreducible amount of computation—that separates our actual behavior from them.

At various times in the history of exact science, people have thought there might be some complete predictive theory of human behavior. And what we can now see is that in a sense there's a fundamental reason why there can't be. So the result is that at some level to know what will happen, we just have to watch and see history unfold.

Of course, as a practical matter, what we can "watch" is becoming more and more extensive by the year. It used to be that very little of history was recorded. A whole civilization might leave only a few megabytes, if that, behind.

But digital electronics has changed all of that. And now we can sense, probe and record immense details of many things. Whether it's detailed images of the Earth's surface, or the content of some network of human communications, or the electrical impulses inside a brain. We can store and retrieve them all. Increasingly, we'll even be able to go back and reproduce the past. A few trace molecules in some archaeological site, extrapolated DNA for distant ancestors and so on. I expect we'll be able to read the past of almost any solid surface. Every time we touch something, we disturb a few atoms. If we repeat it enough, we'll visibly wear the solid down. But one day, we'll be able to detect just that first touch by studying the whole pattern of atoms on the surface.

There's a lot one could imagine knowing about the world. And I think it's going to become increasingly possible to find it out, once one asks for it. Yet just how the sensors and the systems they are able to sense will relate is an interesting issue.

One of the consequences of my Principle of Computational Equivalence is that sophisticated computation can happen in a tremendous range of systems—not just brains and computers, but also all sorts of everyday systems in nature. And no doubt our brains—and current computers—are not especially efficient vehicles for achieving computation. And as our technology gets better, we'll be able to do computation much better in other media. Making computation happen, for example, at the level of individual

molecules in materials.

So it'll be a peculiar picture: the computations we want to do happening down at an atomic scale. With electrons whizzing around—pretty much just like they do anyway in any material. But somehow in a pattern that is meaningful with respect to the computations we want to do.

Now perhaps a lot of the time we may want to do "pure computation"—in a sense just purely "think". But sometimes we'll want to interact with the world—find out knowledge from the world. And this is where our sensors come in. But if they too are operating at an atomic scale, it'll be just as if some clump of atoms somewhere in a material is affecting some clump of atoms somewhere else—again pretty much just like they would anyway.

It's in a sense a disappointing picture. At the end of all of our technology development, we're operating just like the rest of the universe. And from the outside, there's nothing obvious we've achieved. You'd have to know the history and the context to know that those electrons whizzing around, and those atoms moving, were the result of the whole rich history of human civilization, and its great technology achievements.

It's a peculiar situation. But in a sense I think it reflects a core issue about ultimate knowledge.

Right now, the web contains a few billion pages of knowledge that humans have collected with considerable effort. And one might have thought that it'd be difficult to generate more knowledge.

But it isn't. In a sense that's what Leibniz found exciting about mathematics. It's possible to use it systematically to generate new knowledge. Working out new formulas, or new results, in an essentially mechanical way.

But now we can take that idea much further. We have the whole computational universe to explore—with all possible rules. Including, for example, I believe, the rules for our physical universe. And out in the computational universe, it's easy to generate new knowledge. Just sampling the richness of what even very simple programs can do. In fact, given the idea of computation universality, and especially the Principle of Computational Equivalence, there is a sense in which one can imagine systematically generating all knowledge, subject only to the limitations of computational irreducibility.

But what would we do with all of this? Why would we care to have knowledge about all those different programs out there in the computa-

tional universe? Well, in the past we might have said the same thing about different locations on the Earth. Or different materials. Or different chemicals. But of course, what has happened in human history is that we have systematically found ways to harness these things for our various purposes. And so, for example, over the course of time we have found ways to use a tremendous diversity, say, of possible materials that we can "mine" from the physical world. To find uses for magnetite, or amber, or liquid crystals, or rare earths, or radioactive materials, or whatever. Well, so it will be with the computational universe. It's just starting now. Within *Mathematica*, for example, many algorithms we use were "mined" from the computational universe. Found by searching a large space of possible programs, and picking ones that happen to be useful for our particular purposes. In a sense defining pieces of knowledge from the sea of possibilities that end up being relevant to us as humans.

It will be interesting to watch the development of technology—as well as art and civilization in general—and to see how it explores the computational universe of possible programs. I'm sure it'll be not unlike the case of physical materials. There'll be techniques for mining, refining, combining. There'll be "gold rushes" as particular rich veins of programs are found. And gradually the domain of what's considered relevant for human purposes will expand to encompass more and more of the computational universe.

But, OK, so there is all sorts of possible knowledge out there in the computational universe. And gradually our civilization will make use of it.

But what about particular knowledge that we would like to have, today? What about Leibniz's goal of being able to answer all human questions by somehow systematizing knowledge?

Our best way of summarizing and communicating knowledge tends to be through language. And when mathematics became formalized, it did so essentially by emulating the symbolic structure of traditional human natural language. And so it's interesting to see what's happened in the systematization of mathematics.

In the early 1900s, it seemed like the key thing one wanted to do was to emulate the process of mathematical proof. That one wanted in effect to find nuggets of truth in mathematics, represented by proofs. But actually, this really turned out not to be the point. Instead, what was really important about the systematization of mathematics was that it let one specify calculations. And that it let one systematically "do" mathematics

automatically, by computer, as we do in *Mathematica*.

Well, I think the same kind of thing is going to be true for ordinary language. For centuries, people have had the idea of somehow formalizing language: nowadays, of making something like a computer language that can talk about everyday issues. But the question is: what is supposed to be its purpose? Is its purpose—like the formalizations of mathematical proof—to represent true facts in the world? If so, then to derive useful things one has to have some kind of inferencing mechanism—something that lets one go from some facts, or some theorems, to others.

And as for proof-based mathematics, there is certainly something to be done here. But I think the much more important direction is the analog of calculation-based mathematics. Somehow to take a formalization of everyday discourse, and calculate with it. What could this mean? What is the analog of taking a mathematical expression like 2+2 and evaluating it?

In a sense it is to take a statement, and work out statements that are somehow the result of it.

Out in the computational universe, there are lots of systems and processes. And while computational irreducibility may force us to use explicit simulation to work out their results, we have a definite procedure for doing what we can do.

The issue, however, is to connect this "vast ocean of truth" with actual everyday questions. The problem is not so much how to answer questions, as how to ask them. As our ability to set up more and more elaborate networks of sensors increases, there will be a new approach. We will be able to take "images" of the world, and directly map them onto systems and processes in the computational universe. And then find out their results not by watching the systems in nature, but by abstractly studying their analogs in the computational universe.

Perhaps one day the analog of human discourse will operate more at the level of such "images". But for now traditional language is our primary means of communicating ideas and questions. It is in a sense the "handle" that we must use to specify aspects of the computational universe that we want to talk about.

Of course, language evolves as different things become common to talk about. In the past, we would have had no words for talking about nested patterns. But now we just describe them as "nested" or "fractal".

But if we just take language as it is, it defines a tiny slice of the computational universe. It is in many ways an atypical slice. For it is highly weighted towards computational reducibility. For we, as humans, tend to concentrate on things that make sense to us, and that we can readily summarize and predict. So at least for now only a small part of our language tends to be devoted to things we consider "random", "complex", or otherwise hard for us to make sense of.

But if we restrict ourselves to those things that we can describe with ordinary language, how far can we go in our knowledge of them? In most directions, computational irreducibility is not far away—providing in a sense a fundamental barrier to our knowledge. In general, everyday language is a very imprecise way to specify questions or ideas, being full of ambiguities and incomplete descriptions. But there is, I suspect, a curious phenomenon that may be of great practical importance. If one chooses to restrict oneself to computationally reducible issues, then this provides a constraint that makes it much easier to find a precise interpretation of language. In other words, a question asked in ordinary language may be hard to interpret in general. But if one chooses to interpret it only in terms of what can be computed—what can be calculated—from it, it becomes possible to make a precise interpretation.

One is doing what I believe most of traditional science has done: choosing to look only at those parts of the world on which particular methods can make progress. But I believe we are fairly close to being able to build technology that will let us do some version of what Leibniz hoped for. To take issues in human discourse, and when they are computable, compute them.

The web—and especially web search—has defined an important transition. It used to be that static human knowledge—while in principle accessible through libraries and the like—was sufficiently difficult to access that a typical person usually sampled it only very sparingly. But now it has become straightforward to find "known facts". Using the "handle" of language, we just have to search the web for where those facts are described.

But what about facts that are not yet "known"? How can we access those? We need to create the facts, by actual computation. Some will be fraught with computational irreducibility, and in some fundamental sense be inaccessible. But there will be others that we can access, at least if we can find a realistic way for us humans to refer to them.

Today there is a certain amount of computation about the world that we routinely do. Most of it is somehow done automatically inside devices that use their own sensors to find out about the world—automatic cameras, GPS devices, and so on. And there is a small subset of people—physicists, engineers, and the like—who fairly routinely actually do computations about the world. But despite the celebrated history of exact science, few people have direct access to the computations it implies can be done.

I think this will change. Part of the change is already made possible with *Mathematica* as it is today. But another part is coming, with new technology that we are working to build. And the consequence of it will be something that I believe will be of quite fundamental importance. That we will finally be able routinely to access what can be computed about our everyday world. In a sense to have ultimate access to the knowledge which it is possible to get.

Extrapolating from Leibniz, we might have hoped that we would be able to get ultimate knowledge about everything. Somehow with our sophistication to be able to work out everything about the world, and what can happen in it. But we now know that this will never be possible. And indeed, from looking at the computational universe, it becomes clear that there is a lot in the world that we will never be able to "unravel" in this way. In some ways it would have been disappointing if this had been so. For it would mean that our world could somehow be simplified. And that all the richness of what we actually see—and the actual processes that go on in nature—would be unnecessary. And that with our ultimate knowledge we would be able to work out the true "results" of our universe, without going through everything that actually happens in the universe.

As it is, we know that this is not the case. But increasingly we can expect that whatever knowledge can in principle be obtained, we will actually be able to obtain. To fully harness the concept of computation, and to integrate it into the future of our civilization.

Chapter 23

An Enquiry Concerning Human (and Computer!) [Mathematical] Understanding

Doron Zeilberger[1,2]

Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA; `zeilberg@math.rutgers.edu`

23.1. The Arrogance of Science and Mathematics

Science and mathematics seem to be huge success stories. Hence it is not surprising that most scientists and mathematicians think that science and mathematics are the most secure ways of acquiring *knowledge*, and that all knowledge could, at least in principle, be derived using either the *scientific method*, using *inductive* reasoning, and in the case of *mathematical* knowledge, using *deductive* reasoning.

In the 19th century, people were so impressed with science and mathematics that, starting with Comte, a movement called *positivism*, that tried to apply the so-called scientific method to all domains of inquiry, gained prominence. But then the pendulum swung back, and many objected to what they called the *imperialism of science*, and Comte's *empiric positivism* gave way to Bergson's and others' *metaphysico-spiritual movement*, that emphasized the *heart* rather than the *brain*, and *intuition* rather than *deduction*. A century earlier, German *Romanticism* and *Idealism* were reactions against the *rationalism* of the Enlightment.

More recently, science came under attack by *post-modern* philosophers, and that got some scientists, most notably Alan Sokal, to fight back by making fun of them. Little did Alan know that the *joke is on him*, since while some of the details of the philosophical critiques of science were indeed

[1] Dedicated to my two favorite skeptics: David Hume and Gregory Chaitin.

[2] Supported in part by the NSF.

erroneous and sometimes pure gibberish, the *spirit* of the critiques were very well-founded, since all that they were trying to say was that old standby, that goes back at least to Socrates: *We know that we don't know.*

23.2. Skeptics

I have always admired skeptics, from Pyrrho of Elis all the way to Jacques Derrida. But my two *favorite* skeptics are **David Hume** and **Gregory Chaitin**, who so beautifully and eloquently described the **limits of science** and the **limits of mathematics**, respectively.

23.3. David Hume's Critique of the Scientific Method

According to Bertrand Russell, there is a place in hell for philosophers who *believed* that they solved Hume's *problem of induction*. Of course, no one has yet solved it, and Hume's famous assertion that (physical) *induction*, i.e. *generalizing from* **finitely** *many cases*, has **no (logical) justification whatsoever**, has not yet been rebutted successfully.

Let's cite his doubts about the sun rising tomorrow:

That the sun will not rise tomorrow is no less intelligible a proposition, and implies no more contradiction, than the affirmation, *that it will rise*.

Another, more recent, attack on (physical) induction was launched by Nelson Goodman, who coined the term *grue* for an object that is green before Jan. 1, 2050, and is blue after it. So far all examined emeralds turned out to be green, hence, by (physical, incomplete) induction it is reasonable to state that "all emeralds are green". But, by the same token, so far all emeralds turned out to be grue, so stand by for Jan. 2, 2050, and dear old Goodman predicts that all emeralds will be blue then, since then grue would be blue, and we have such good empirical evidence that they are always grue.

23.4. Greg Chaitin and the Limits of Mathematics

Standing on the shoulders of Gödel, Turing (and Post, Church, Markov and others), Greg Chaitin gave the most *succinct*, *elegant*, and *witty* expression to the **limits** of our mathematical knowledge. It is his immortal **Chaitin's**

Constant, Ω:

$$\Omega := \sum_{p \text{ halts}} 2^{-|p|},$$

where the sum ranges over all *self-delimiting* programs run on some Universal Turing Machine. As Greg puts it so eloquently, Ω is the epitome of *mathematical randomness*, and its digits are beautiful examples of *random mathematical facts*, true for "no reason". It also has the charming property of being *normal to all bases*.

23.5. How Real Is Ω?

There is only one problem with Ω, it is a *real* number! As we all know, but most of us refuse to admit, "real" numbers are **not** real, but purely fictional, since they have **infinitely many** digits, and there is no such thing as infinity. Worse, Ω is **uncomputable**, since we know, thanks to Turing, that there is no way of knowing, *a priori*, whether p halts or not. It is true that many "real" numbers, for example $\sqrt{2}, \phi, e, \pi$ etc., can be *deconstructed* in finite terms, by renaming them 'algorithms', and we do indeed know that these are genuine algorithms since in each specific case, we can prove that any particular digit can be computed in a finite, pre-determined, number of steps. But if you believe in Ω, then you believe in God. God *does* know whether *any* program p will eventually halt or not, because God lives for ever and ever (Amen), and also can predict the future, so for God, Ω is as real as $\sqrt{2}$ or even 2 is for us mere mortals. So indeed, *if* God exists, then Ω exists as well, and God knows all its digits. Just because *we*, lowly mortals, will never know the digits of Ω, is just a reflection on *our* own limitations.

But what if you *don't* believe in God? Or, like myself, does not know for sure, one way or the other?

23.6. Do I Believe in Ω?

Regardless of whether or not God exists, God has no place in mathematics, at least in *my* book. *My* God does not know (or care) whether a program p eventually halts or not. So Ω does **not** exist in my, ultra-finitistic, world-view. But, it does indeed exist as a *symbol*, and as a lovely *metaphor*, so like enlightened 'non-fundamentalist' religious folks, we can still enjoy and believe in the bible, even without taking it literally. I can still love and

cherish and adore Chaitin's constant, Ω, the same way as I enjoy Adam and Eve, or Harry Potter, and who cares whether they are 'real' or 'fictional'.

23.7. Greg Chaitin's Advice About Experimental Mathematics

One interesting moral Greg Chaitin draws from his brainchild, *Algorithmic Information Theory*, and its crown jewel, Ω, is the advice to pursue Experimental Mathematics. Since so much of mathematical truth is inaccessible, it is stupid to insist on finding a proof for every statement, since for one, the proof may not exist (it may well be undecidable), or it may be too long and complicated for us mere humans, and even for our computers. So Greg suggests to take truths that we 'feel' are right (on heuristic or experimental grounds) and adopt them as new 'axioms', very much like physicist use Conservation of Energy and the Uncertainty Principle as "axioms". Two of his favorites are $P \neq NP$ and the Riemann Hypothesis. Of course, by taking these as new 'axioms' we give up on one of the original meanings of the word 'axiom', that it should be 'self-evident', but Hilbert already gave this up by making mathematical deduction into a formal game.

23.8. Stephen Wolfram's Vision

Another, even more extreme, advocate of Experimental Mathematics, is guru Stephen Wolfram, whose *New Kind of Science* and *New Kind of Mathematics* are *completely computer-simulation*-centric. Let's dump traditional equation-centric science and deduction-centric mathematics in favor of doing computer experiments, and watching the output.

23.9. Tweaking Chaitin's and Wolfram's Messages: The Many Shades of Rigor

I admire both Chaitin and Wolfram, but like true visionary prophets, they see the world as *black* and *white*. Since all truths that we humans can know with old-time certainty are doomed to be *trivial* (or else we wouldn't have been able to prove them completely), and conversely, *all* the *deep* results will never be able to be proved by us completely, with traditional standards, they advise us to abandon the old ways, and just learn how to ask our computers good questions, and watch its *numerical* output, and gain *insight* from it.

Things do not have to be so polarized. First, computers can help us find *completely rigorous* proofs, that we humans can never find by ourselves, for example the Four Color Theorem, or the many computer-generated proofs of WZ theory. Second, as I first suggested in my Oct. **1993** *Notices* manifesto, "*Theorems for a Price: Tomorrow's Semi-Rigorous Mathematical Culture*", one can try and prove things semi-rigorously.

So the great insight of Greg Chaitin and Stephen Wolfram can be *fine-tuned* and instead of the *"all or nothing"* mentality regarding rigor, we can introduce a whole *spectrum* of **rigor** and **certainty**.

23.10. The Greek Model for Mathematics and Meta-Mathematics

Meta-mathematics, starting with Frege, continuing through Russell, Whitehead and Hilbert, and culminating in Chaitin and others, has been using the *Euclidean* model of mathematics, trying to *emulate* and *formalize* Euclid's paradigmatic *Elements*. Start with a set of *axioms* (originally required to be *self-evident* but later considered *arbitrary*) and *rules of deduction*, and a notion of *formal proof* and try to derive *all* theorems from the axioms.

Alas, Hilbert's naive dream was shattered by Gödel (and later by Turing, and beautifully explicated by Chaitin) who (allegedly) proved that:

"*There exist true yet unprovable statements*".

Of course, you can *meta-prove* them, but then there would be new statements that you could only meta-meta-prove *ad infinitum*.

23.11. Did Gödel Really Prove That There Exist True yet Unprovable Statements?

Of course not! All his "statements" were *meaningless*!

Every statement that starts : "for every integer n ..." or "there exists an integer n", is completely meaningless, since it tacitly assumes that there are *infinitely* many integers. Of course, there are only finitely many of them, since our *worlds*, both the *physical* and the *mathematical*, are *finite*.

More specifically, the meta-statement:

"P has a proof of length ≤ 1000000 *characters"* does make sense,

and even the meta-statement

"P has a proof of length ≤ googolplex characters" does make sense,

but the "statement":

"P is unprovable"

is the same as the following "statement":

"There does not exist an integer t such that P has a proof of length t characters",

and this "statement" is completely meaningless.

Ditto for the Gödel sentence that is "equivalent" to it, that contains lots of quantifiers.

So, all that Gödel meta-proved was the *conditional* statement:

If "P is unprovable" makes sense and *if* the Gödel sentence makes sense, *then* there exist true yet unprovable statements.

Gödel, being a devout infinitarian platonist, believed in the premises, but I, being a finitistic platonist, see Gödel's proof as a beautiful *reductio* proof that all statements that contain quantifiers are *a priori* meaningless, and only sometimes can be given an *a posteriori* meaning, when interpreted symbolically.

Very often one can *deconstruct* a seemingly 'infinitarian' statement by restating it *symbolically*.

The statement "$n + n = 2n$ for every integer n" is meaningless. It is only true for *every* **finite** integer. It is also true for *symbolic* n.

The statement "every integer has a successor" is meaningless, but one can say that $n + 1$ is the *symbolic* successor of n. Gödel's 'true' yet unprovable statements are simply statements that may not be resurrected for symbolic n. *A priori*, the statement "*there are infinitely many twin primes*" makes no sense, and neither does "*there are infinitely many primes*". *A posteriori* the latter can be made to make sense, by showing the validity of the algorithm implicit in Euclid's 2300-year-old proof, that manufactures 'yet another prime' (but symbolically!). I am sure that the twin-prime conjecture is also true, since it would turn out to be true for symbolic n. If $A(n)$ is the number of twin-prime pairs $\leq n$, then, some future sieving inequality

(that will be found by computer!), will imply that

$$A(n) \geq C_1 \frac{n}{(\log n)^2},$$

for *symbolic* n, and specific C_1, that would contradict the symbolic inequality $A(n) \leq C_2$, C_2 being a (symbolic!) constant.

23.12. The Chinese-Indian-Sumerian-Egyptian-Babylonian Model for Doing Mathematics

Euclid ruined mathematics by introducing that pernicious *axiomatic method* and making mathematics *deduction-centric*. But for thousands of years before Euclid, mathematics has been pursued *empirically* and *experimentally* and was *induction-centric*. It was what Richard Feynman called *Babylonian-style* mathematics. The reason Feynman liked it so much is that not only was it empirical, but it was also **algorithmic**.

23.13. Formalizing Algorithms: Turing Machines

Algorithms existed for at least five thousand years, but people did not know that they were algorithmizing. Then came Turing (and Post and Church and Markov and others) and *formalized* the notion. In the case of Turing, he introduced *Turing machines*. Of course, given an algorithm, it is nice to know that it is indeed an algorithm, and not just a Turing machine, in other words, that it *halts*. But the question "does T halt" is also meaningless. On the other hand: "does T halt in $\leq$ 1000 years" does make sense. So, by hindsight, just like in Gödel's case, it is not at all surprising that there in no decision algorithm for the halting problem. It was a stupid (in fact, worse, meaningless) question to begin with, and Turing just meta-proved that it was indeed very stupid to expect such an algorithm, and there is no way to make sense of it even *a posteriori*.

23.14. The Problem with the Chaitin-Kolmogorov Definition of Program-Size Complexity and Randomness

Greg Chaitin, and independently Andrey Kolmogorov and Ray Solomonoff, famously defined *program-size complexity* of a (finite or infinite) string as the *length of its shortest description* in some **fixed description language**. Now, that *description language* could be taken to be English, French, Hebrew, Spanish, or Chinese. But *natural* languages are notoriously fuzzy, and

may be good media for love songs, but not for mathematics and computer science. The *lingua-franca* of theoretical computer science is the Turing machine. There are also numerous equivalent models, that are sometimes easier to work with. But even this is too vague, since we can't tell, thanks to Turing, whether our TM would halt or not, in other words whether it is a *genuine* algorithm or just an *algorithm wannabe*. Furthermore, even if it *does* halt, if my super-short computer program would take *googolplex* to the power *googolplex* years to generate my sequence, it can't do me much good. It is true that for *aesthetic* reasons, Greg Chaitin refused to enter time into his marvelous theory, and he preempted the criticism by the disclaimer that his theory is 'useless for applications'. But, I, for one, being, in part, a *naturalist*, find it hard to buy this nonchalance. Life is finite (alas, way too finite), and it would be nice to reconcile time-complexity with program-size complexity. Anyway, using *Turing machines* or any of the other computational models, for which the halting program is undecidable, makes this notion *meaningless*. Of course it has a great *metaphoric* and *connotative* meaning!

So the notion of *Turing machine*-computable is way too general. Besides the Greek model, adopted by mathematicians and meta-mathematicians alike does not represent how most of mathematics is done in **practice**.

Most of mathematics, even logic, is done within narrow *computational frameworks*, sometimes explicit, but more often implicit. And what mathematicians do is *symbol-crunching* rather than *logical deduction*. Of course, formal logic is just yet another such symbolic-computational framework, and *in principle* all proofs can be phrased in that language, but this is *unnatural*, *inefficient*, and worse, sooo **boring**.

Let's call these computational frameworks **ansatzes**. In my humble opinion, mathematics should abandon the Greek model, and should **consciously** try to **explicate** more and more new ansatzes that formerly were only implicit. Once they are made explicit, one can teach them to our computers and do much more than any human.

23.15. The Ansatz Ansatz

Indeed, lots of mathematics, as it is *actually* practiced today, can be placed within well-defined *computational frameworks*, that are provably *algorithmic* and, of course, *decidable*. Sometimes the practitioners are aware of this, and in that case 'new' results are considered routine. For example,

the theorem

$$198765487 \cdot 198873987 \;=\; 39529284877686669,$$

is not very exciting today, since it belongs to the well-known class of **explicit arithmetical identities**.

On the other hand, the *American Mathematical Monthly* still publishes papers today in Euclidean Geometry, that, thanks to René Descartes, is reducible to *high-school algebra*, that is also routinely provable, of course in principle, but today also in practice, thanks to our powerful computer algebra systems.

The fact that multiplication identities are routinely provable is at least 5000-years old, and the fact that theorems in Plane Geometry are routinely-provable is at least 250-years old (and 40-year old in practice), but the fact that an identity like

$$\sum_{k=-n}^{n} (-1)^k \binom{2n}{n+k}^3 = \frac{(3n)!}{n!^3},$$

discovered, and first proved in 1904 by Dixon, is also routinely provable, is only about 16-years old, and is part of so-called *Wilf-Zeilberger* Theory.

In each of these cases it is *nowadays* routine to prove an identity of the form $A = B$, since there is *a canonical form* algorithm $A \to c(A)$, and all we have to do is check that $c(A) = c(B)$. In fact, to prove that $A = B$, it suffices to have a *normal-form* algorithm, checking that $A - B$ is 'equivalent' to 0.

But before we can prove a statement of the form $A = B$, we have to find an *appropriate* **ansatz** to which they both belong.

At this time of writing, there are only a few explicitly known ansatzes. Let's first review one of my favorites.

23.16. The Polynomial Ansatz

David Hume is right that there is no formal, watertight, proof that the sun will rise tomorrow, since the Boolean-valued function

$$f(t) := evalb(The \quad Sun \quad Will \quad Rise \quad At \quad Day \quad t),$$

has not yet been proved to belong to any known ansatz. Indeed, we now know, that for $t >> 0$, $f(t)$ is false, because the Sun will swallow Planet Earth, so all we can prove are vague probabilistic statements for small t

(e.g. for t = tomorrow).

The Clay Foundation is also right that there is not yet a formal, watertight, proof, of the Riemann Hypothesis, even though Andrew Odlyzko and Herman te Riele proved that the first ten billion, or whatever, complex zeros of $\zeta(s)$ lie on the critical line. This is because the sequence

$$f(n) := Re(z_n),$$

where z_n is the n^{th} complex root of $\zeta(s) = 0$, has not yet been proved to belong to any known ansatz.

However, the following proof of the lovely identity

$$\sum_{i=1}^{n} i^3 = \left(\sum_{i=1}^{n} i \right)^2 ,$$

is perfectly rigorous.

Proof: True for $n = 0, 1, 2, 3, 4$ (check!), hence true for all n. QED

In order to turn this into a full-fledged proof, all you have to do is mumble the following incantation:

Both sides are polynomials of degree ≤ 4, hence it is enough to check the identity at five distinct values.

23.17. An Ansatz-based Chaitin-Kolmogorov Complexity

So let's define the *complexity* of an infinite (or finite) sequence always *relative* to a given *ansatz*, assuming that it indeed belongs to it. So our *descriptive language* is much more modest, but we can always determine its complexity, and everything is decidable. It does not have the *transcendental beauty* and *universal insight* of Chaitin's Algorithmic Information Theory, but on the other hand, we can always decide things, and nothing is unknowable (at least in principle).

23.18. It All Depends on the Data Structure

Even within a specific ansatz, there are many ways of representing our objects. For example, since a polynomial P, of degree d is determined by its values at any $d+1$ values, we can represent it in terms of a finite sequence $[P(0), \ldots, P(d)]$ that requires $d+1$ "bits" (units of information). Of course, we can also express it in the usual way, as a linear combination of the powers $\{1, n, n^2, \ldots, n^d\}$, or in terms of any other natural base, for

example $\{\binom{n}{k}, k = 0 \ldots d\}$. Each of these data structures require $d + 1$ "bits", in general, but in specific cases we can sometimes *compress* in order to get lower complexity. For example it is much shorter to write n^{1000} then to write $[0, 1, 2^{1000}, \ldots, 1000^{1000}]$ (without the "...", and spelled-out).

23.19. The Strong N_0 Property

An ansatz has the Strong N_0 property, if given any two sequences, A, B, within that ansatz, in order to prove that $A(n) = B(n)$ (for all n), there exists an *easily computable* (say polynomial-time in the maximal size of A and B) number $N_0 = N_0(A, B)$ such that in order to prove that $A(n) = B(n)$ for all n, it suffices to prove it for any N_0 *distinct* values of n.

The *iconic example* of an ansatz having the strong N_0 property, already mentioned above, is the set of *polynomials*. For polynomials $P(x)$ of a single variable, $N_0(P(x))$ is $deg\, P + 1$. For a polynomial $P(x_1, \ldots, x_n)$ of degree d, $N_0(P)$ is $\binom{d+n}{n}$.

23.20. The Weak N_0 Property

An ansatz has the Weak N_0 property, if given any two sequences, A, B, within that ansatz, in order to prove that $A(n) = B(n)$ (for all n), there exists an *easily computable* (say polynomial-time in the maximal size of A and B) number $N_0 = N_0(A, B)$ such that in order to prove that $A(n) = B(n)$ for all n, it suffices to prove it for the *first* N_0 values of n: $n = 1, n = 2, \ldots, n = N_0$.

A simple example of an ansatz that has the weak, but not the strong, N_0 property, are periodic sequences. If two sequences are known *a priori* to have periods d_1 and d_2, then if they are equal for the **first** $max(d_1, d_2)$ values, then they are identically equal. But the two sequences $f(n) := 1$ and $g(n) := (-1)^n$ coincide at infinitely many places (all the even integers), yet the two sequences are not identically equal.

23.21. Back to Science: The PEL Model

In Hugh G. Gauch's excellent book on the Scientific method "*Scientific Method in Practice*", he proposes the *PEL model*, PEL standing for "Presupposition, Evidence, Logic". So Hume's objection disappears if we are willing to concede that science is *theory laden*, and we have lots of presup-

positions, both explicit and implicit.

Now the analog of presupposition in mathematics is ansatz. If we make the reasonable presupposition that the function

$$f(t) := evalb(The \quad Sun \quad Will \quad Rise \quad At \quad Day \quad t) \quad ,$$

belongs to the *constant* ansatz (at least for the next 100000 years), then checking it in *just one point*, say $t = today$, proves that the sun will indeed rise tomorrow.

On the other end, to prove that all emeralds are grue, presupposes that the color of emeralds belong to the *piece-wise constant* ansatz, since the notion of 'grue' belongs to it. In that case, $N_0 > 2050$, so indeed checking it for many cases but before 2050, does not suffice, even non-rigorously, to prove that all emeralds are grue.

23.22. The Probabilistic N_0 Property

Sometimes N_0 is way too big, in other words, to get *complete certainty* will take too long. Then you might want to consider settling for $N_0(p)$.

An ansatz that has the probabilistic N_0-property, is one for which, in order to prove that $A \equiv B$, with probability p, there exists an *easily computable* (say polynomial-time in the maximal size of A and B) number $N_0(p) = N_0(A, B, p)$ such that in order to prove that $A(n) = B(n)$ for all n with probability p, it suffices to prove it for *any* $N_0(p)$ randomly chosen values of n.

The celebrated Schwartz-Zippel theorem establishes that multi-variable polynomials satisfy the $N_0(p)$ property (in addition to having the N_0 property, of course), and that $N_0(.9999999)$ is much smaller than $N_0(1)$, so it is stupid to pay for full certainty.

23.23. An Embarrassing Paper of Mine

Can you envision a professional mathematician publishing a paper entitled "A bijective proof of $10 \times 5 = 2 \times 25$", by concocting a nice bijection? Of course not! Today, all explicit arithmetical identities are known to be routinely provable.

Yet something analogous happened to me. In my web-journal, I pub-

lished a paper that found an 'elegant' combinatorial proof of the identity

$$\sum_{i=0}^{2n} \binom{2n}{i} F_{2i} = 5^n F_{2n},$$

where F_n are the Fibonacci numbers defined by $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$ $(n \geq 2)$. It was in response to a challenge by Arthur Benjamin and Jennifer Quinn, posed in their delightful books "*Proofs that really count*".

As "elegant" and "insightful" as my proof may have been, in Occam's and Chaitin's sense, the following proof is much more elegant.

Proof: Both sides are sequences that are solutions of second-order linear recurrence equations with constant coefficients. Hence, to prove that they coincide for all $n \geq 0$, it suffices to check that they coincide for $n = 0, 1, 2, 3$. Now just check that indeed

$n = 0:$ $\quad 1 \cdot 0 = 0,$

$n = 1:$ $\quad 1 \cdot 0 + 2 \cdot 1 + 1 \cdot 3 = 5 \cdot 1$

$n = 2:$ $\quad 1 \cdot 0 + 4 \cdot 1 + 6 \cdot 3 + 4 \cdot 8 + 1 \cdot 21 = 5^2 \cdot 3,$

$n = 3:$ $\quad 1 \cdot 0 + 6 \cdot 1 + 15 \cdot 3 + 20 \cdot 8 + 15 \cdot 21 + 6 \cdot 55 + 1 \cdot 144 = 5^3 \cdot 8.$

QED

Of course, we have to *justify* the claims that both sides are solutions of linear recurrence equations with constant coefficients (by the way, such sequences are called C-finite), of second order. But these follow from the following easy claims, that can be proved *once and for all*, using elementary linear algebra (you do it!).

Claim 1: If a_n is a solution of a linear recurrence equation with constant coefficients of order d, then for any positive integer L

$$b_n := a_{nL},$$

is likewise a solution of a (different) linear recurrence equation with constant coefficients of order d.

Claim 2: If a_n is a solution of a linear recurrence equation with constant coefficients of order d, then its **binomial transform**,

$$b_n := \sum_{i=0}^{n} \binom{n}{i} a_i,$$

is likewise a solution of a (different) linear recurrence equation with constant coefficients of order d.

Claim 3: if a_n satisfies such an order-d recurrence, so does $k^n a_n$.

Claim 4: The algebra of C-finite sequences has the weak N_0-property, and two C-finite sequences of order $\leq d$ are identical if they are identical for $0 \leq n \leq 2d-1$.

For more complicated identities involving C-finite sequences the following claim is need.

Claim 5: If a_n and b_n are C-finite sequences of orders d_1 and d_2, then $a_n + b_n$ are $a_n b_n$ are C-finite of orders $\leq d_1 + d_2$ and $\leq d_1 \cdot d_2$ respectively.

Since any polynomial sequence is C-finite (a polynomial of degree d satisfies the recurrence
$(N-1)^{d+1} f(n) = 0$, where N is the forward-shift operator), it follows that the ansatz of C-finite sequences is a superset of the polynomial ansatz. The next ansatz is even bigger, and contains that of C-finite sequences.

23.24. The Schützenberger Ansatz

If the generating function of a sequence $\{a(n)\}_{n=0}^{\infty}$,

$$\phi(x) = \sum_{n=0}^{\infty} a(n) x^n,$$

satisfies a polynomial equation:

$$P(\phi(x), x) = 0,$$

then it is called an *algebraic* formal power series. I call it the Schützenberger Ansatz, since it was Marco Schützenberger's favorite ansatz, and has gotten lots of attention by his illustrious disciple Xavier Viennot and Viennot's disciple the brilliant Mireille Bousquet-Mélou, and numerous others at the *école bordelaise*.

This is also an algebra, and every identity is decidable, and it, too, has the weak N_0 property.

23.25. Solving Functional Equations Empirically (Yet Rigorously!)

In many combinatorial problems, one is interested in a formal power series $F(x, y; t)$ that satisfies a functional equation of the form

$$A(x,y,t)F(x,y;t) + B(x,y,t)F(0,y;t) + C(x,y,t)F(x,0;t)$$

$$+D(x,y,t)F(0,0;t) = E(x,y,t) \quad , \qquad (FunEq)$$

where A, B, C, D, E are polynomials in (x, y, t). Such an equation can be used to crank out the Maclaurin expansion of $F(x, y; t)$ to any desired order.

Often, we are really only interested in $\phi(t) := F(0, 0; t)$, that sometimes, surprisingly, happens to satisfy some nice algebraic equation $P(t, \phi(t)) = 0$, for no apparent reason, and the challenge is to prove that fact. $(FunEq)$ can't be used directly, since it involves (x, y, t) not just t, and plugging-in $x = 0, y = 0$ in $(FunEq)$ usually yields the fact $0 = 0$, that while true, is far from new, and does not help us with the conjecture at hand.

While we are **not** guaranteed, *a priori*, that this is the case, it is still worthwhile to try to conjecture that not only $\phi(t)$ is algebraic, but so is the full $F(x, y; t)$, i.e. there exists a polynomial Q, $Q(F, x, y, t)$ such that

$$Q(F(x,y;t),x,y,t) \equiv 0. \qquad (AlgEq)$$

This Q can be found, empirically for now, by the method of *undetermined coefficients*. Use $(FunEq)$ to crank out the first 1000 or whatever terms of F, call the truncated version $\widetilde{F}$; let Q be a generic polynomial of four variables of a guessed degree d, with undetermined coefficients; ask the computer to compute $Q(\widetilde{F}(x, y; t), x, y, t)$, set the first 1000 terms to 0, get a huge system of equations for the undetermined coefficients, and solve them. If there is a non-zero solution, then it is great news! Otherwise, make d bigger, or give up.

Once we (or rather our computer) conjectured such a general algebraic equation, how do we prove it rigorously?

We have to prove that $(FunEq)$ implies $(AlgEq)$. By uniqueness, we can prove that $(AlgEq)$ implies $(FunEq)$. Defining $G(x, y; t)$ to be the unique solution of

$$Q(G(x,y;t),x,y,t) \equiv 0, \qquad (AlgEq)$$

it follows that $G(x, 0; t)$, $G(0, y; t)$, $G(0, 0; t)$ are all algebraic:

$$Q(G(x,0;t),x,0,t) \equiv 0, \qquad (AlgEq')$$

$$Q(G(0,y;t),0,y,t) \equiv 0, \qquad (AlgEq'')$$

$$Q(G(0,0;t),0,0,t) \equiv 0. \qquad (AlgEq''')$$

Now

$$H(x,y,t) := A(x,y,t)G(x,y;t) + B(x,y,t)G(0,y;t) + C(x,y,t)G(x,0;t)$$

$$+D(x,y,t)G(0,0;t) - E(x,y,t)$$

is also algebraic and using the "Schützenberger calculator" one can find an equation satisfied by it, and prove that H is identically 0, and by uniqueness, $F = G$.

Now plugging-in $x = 0, y = 0$, into the *now-proved* algebraic equation $Q(F(x,y,t),x,y,t) = 0$, would yield a rigorous proof of the conjectured algebraic equation for $\phi(t) = F(0,0,t)$, namely $Q(\phi(t),0,0,t) \equiv 0$.

The downside in the above empirical (yet *a posteriori* rigorous!) approach, is that the computations required to conjecture Q are very heavy, and for all but the simplest problems, the above method is beyond today's computers. Also, in practice it is more efficient to first conjecture algebraic equations for $F(x,0;t)$ and $F(0,y;t)$ and use the "calculator" to derive what the algebraic equation for the $F(x,y;t)$ should be.

A yet more powerful ansatz, that contains all the preceding ones considered so far is the *Holonomic Ansatz*, that is my absolute personal favorite.

23.26. The Holonomic Ansatz

A sequence $\{a(n)\}$ is holonomic if it satisfies a *linear recurrence equation* with **polynomial** coefficients. The sum and product of holonomic sequences is again holonomic, and one has a 'holonomic calculator' (The Salvy-Zimmerman Maple package `Gfun`).

Introducing the shift operator $Nf(n) := f(n+1)$, one can define a holonomic sequence in terms of its *annihilating operator* $P(N,n)$ and the initial conditions.

A discrete function of several variables $a(n_1,\ldots,n_k)$ is holonomic if for each variable n_i there is an annihilating operator $P_i(n_1,\ldots,n_k\,;\,N_i)$. This is the basis for so-called Wilf-Zeilberger theory and it is not only closed with respect to addition and multiplication, but also with respect to sums. For example, if $F(n,k)$ is holonomic, then $a(n) := \sum_k F(n,k)$ is holonomic as well.

23.27. Functional Equations and Holonomic Functions

Analogous remarks about the interface between functional equations and algebraic formal power series apply for finding a possible holonomic representation for a formal power series given as a solution of a functional equation.

23.28. In Search of New Ansatzes

The above ansatzes are just some of those known today. I am sure that the future will bring lots of new ansatzes that will trivialize and routinize large parts of mathematics.

23.29. Pólya's Heuristic Applied to Computer Generated Mathematics

One principle George Pólya was very fond of was "finding the right generalization". Suppose that you conjecture that $A(n) = B(n)$ but you can only prove it for $1 \leq n \leq 7$, because it takes too much time and space to verify it for $n = 8$ and beyond. Of course you can't generalize from seven cases! But if you can find *two-parameter* objects $C(m, n)$ and $D(m, n)$ such that $A(n) = C(n, n)$ and $B(n) = D(n, n)$, and you can prove that $C(n, m) = D(n, m)$ for $1 \leq n \leq 7$, for *all* $m \geq 0$, then the conjecture C=D is true for infinitely many cases, so C=D is very plausible, and hence A=B.

23.30. A Very Simple Toy Example

Let $A(n)$ be the number of words in the alphabet $\{1, 2\}$ with exactly n 1's and exactly n 2's.

By direct enumeration you find that

$$A(0) = 1, A(1) = 2, A(2) = 6,$$

$$A(3) = 20, A(5) = 252, A(6) = 924,$$

and this leads you to conjecture that $A(n) = B(n)$ where $B(n) = (2n)!/n!^2$.

How would you go about proving this conjecture?

Let's consider the *more general* problem of finding $C(m, n)$, the number of words in the alphabet $\{1, 2\}$ with exactly m 1's and exactly n 2's.

Then $C(m,0)=1$, and you have the following recurrence, easily derived by looking at the number of 2's to the left of the rightmost 1:

$$C(m,n)=\sum_{i=0}^{n} C(m-1,i), \tag{1}$$

from which you can easily deduce the following special cases:

$$C(m,1)=\binom{m+1}{1}, \quad C(m,2)=\binom{m+2}{2}, \quad C(m,3)=\binom{m+3}{3},$$

that naturally leads to the conjecture $C(m,n)=D(m,n)$, where $D(m,n)=\binom{m+n}{n}$. It can be verified for $n \le 10$ easily by using (1) with specific n but general m, by *only* using polynomial summation. **Now** the more general statement, $C=D$, is much more plausible. Besides, this more general conjecture is much easier to prove, since you have more elbow room, and it is easy to prove that both $X=C$ and $X=D$ are solutions of the linear *partial recurrence boundary-value problem*:

$$X(m,n)=X(m-1,n)+X(m,n-1), \quad X(m,0)=1, \quad X(0,n)=1.$$

So in this case finding the right generalization first made our conjecture much more plausible, and then also made it easy to prove.

23.31. How to Do It the Hard Way

In order for you to appreciate how much trouble could be saved by introducing a more general conjecture, let's do it, the **hard way**, sticking to the original one-parameter conjecture.

Let $b(n)$ be the number of words in $\{1,2\}$ with exactly n 1's and with exactly n 2's such that in addition , for any proper prefix, the number of 1's always exceeds the number of 2's. Analogously, Let $b'(n)$ be the number of words in $\{1,2\}$ with exactly n 1's and with exactly n 2's such that in addition the number of 2's always exceeds the number of 1's except at the beginning and end. By symmetry $b(n)=b'(n)$.

Then we have the non-linear recurrence

$$a(n)=\sum_{m=0}^{n-1} a(m)(b(n-m)+b'(n-m))=2\sum_{m=0}^{n} a(m)b(n-m). \tag{2}$$

obtained by looking at the longest prefix with the same number of 1's and 2's. Also, using a standard combinatorial argument, $b(n)$ can be shown to

satisfy a non-linear recurrence

$$b(n) = \sum_{m=1}^{n-1} b(m)b(n-m),$$

from which you can crank out many values of $b(n)$, that in turn, enable you to crank out many values of $a(n)$, and make your conjecture much more plausible. Using the above non-linear recurrence, you can generate the first few terms of the sequence $\{b(n)\}_{n=1}^{\infty}$: $1, 1, 2, 5, 14, 42, 132, \ldots$, and easily guess that $b(n) = \frac{(2n-2)!}{(n-1)!n!}$, and to prove it rigorously, all you need is verify the binomial coefficient identity

$$\frac{(2n-2)!}{(n-1)!n!} = \sum_{m=1}^{n-1} \frac{(2m-2)!}{(m-1)!m!} \cdot \frac{(2n-2m-2)!}{(n-m-1)!(n-m)!},$$

that can be done automatically with the WZ method, and then prove the identity

$$\binom{2n}{n} = 2\sum_{m=0}^{n-1} \binom{2m}{m} \cdot \frac{(2n-2m-2)!}{(n-m-1)!(n-m)!},$$

that is likewise WZable.

Note: One can also do it, of course, with generating functions, staying within the Schützenberger ansatz rather than the holonomic anstaz. But it is still much harder than doing it via the 2-parameter generalization discussed above.

23.32. Pólya's Ode to Incomplete Induction

In Polya's masterpiece on the art of mathematical *discovery*, "**Induction and Analogy in Mathematics**" he lauded the use of incomplete induction as a powerful *heuristic* for discovering mathematical conjectures, and as a tool for discovering possible proofs. In particular he cites approvingly the great Euler who conjectured, long before he had a formal proof, many interesting results. For example:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6},$$

that he verified numerically to six decimal places, noting that this implies that the probability that the left side and right hand side coincide by accident is less than one in a million. Many years later he found a complete

proof, but he first had a "cheating proof" that proceeded by pretending that infinite products are like polynomials. Another notable example was the pentagonal number theorem, that he conjectured, and deduced important consequences from, based on expanding it to eighty terms. Only 25 years did he find a formal proof.

Undoubtedly, the greatest conjecturer of all time was Srinivasa Ramanujan, who only needed very few special cases to formulate a conjecture, and was very seldom wrong.

23.33. The Law of Small Numbers

The conventional wisdom *against* the use of incomplete induction is called the *law of small numbers*, and there are many cases, many of them collected by Richard Guy in his two *Monthly* papers about that "Law", that should be cautionary tales against having insufficient data and "jumping to conclusions".

We all know the joke about the mathematician, physicist, and engineer, the mathematician saying "1 is a prime, 3 is a prime, 5 is a prime, 7 is a prime", hence all odd numbers are primes.

Deeper victims of the law of small numbers were Margie Readdy and Richard Ehrenborg who conjectured that the number of up-down involutions of length $2k$ is $k!$, based on data for $k = 1, 2, 3, 4, 5$. This was disproved, by Shalosh B. Ekhad, for $k = 6$ (and beyond). Later Richard Stanley explained why the sequence starts out like that.

Sometimes even 9 terms do not suffice. Neil Sloane, the great master-sequencer, pointed my attention to sequences **A0060041** and **A076912**, in his legendary **database**, that are known to be equal up to $n = 9$, but are believed to disagree for $n = 10$.

But the greatest source of such horror stories is *number theory*.

We all know how the great Fermat goofed when he conjectured that $2^{2^n} + 1$ is always prime, based on the five cases $n = 0, 1, 2, 3, 4$.

Another scary story involves a stronger version of the Riemann Hypothesis, due to Mertens. Recall that the Riemann Hypothesis is equivalent to the statement that the partial sums of the Möbius function:

$$M(n) := \sum_{i=1}^{n} \mu(i),$$

satisfy

$$|M(n)| \leq C(\epsilon)n^{1/2+\epsilon}.$$

Mertens, in 1897, conjectured the stronger conjecture that $|M(n)| \leq n^{1/2}$, and it was verified for n up to a very large number. Yet in 1985, Andrew Odlyzko and Herman te Riele disproved it.

Another notorious example concerns the Skewes Number, that is the smallest n for which $\pi(n)$, the number of prime numbers $\leq n$, is larger than $li(n)$, the logarithmic integral. No one knows its exact value, but it seems to be very large.

23.34. Inequalities vs. Equalities

By hindsight, it is not surprising that both $\pi(n) < li(n)$ and $|M(n)| < \sqrt{n}$ turned out to be false, even though they are true for so many values of n. First, prime numbers are very hazardous, and since often we have $\log\log$ and $\log\log\log$ showing up, it is reasonable to suspect that what seems large for us is really peanuts. But a better reason to distrust the ample empirical evidence is that *inequalities* need much more evidence than *equalities*.

A trivial example is the following. To prove that $P(x) = 0$ for a polynomial P of degree $\leq d$ (say given in some complicated way that is not obviously 0, for example $(x^4+1)(x+1) - x^5 - x^4 - x - 1$) it suffices to check $d+1$ special cases, but consider the "conjecture"

$$\frac{x}{10000000000000} - 1 < 0.$$

The left side is a polynomial of degree 1 in x, and the "conjecture" is true for the first 10000000000000 integer values of x, yet, of course, it is false in general.

23.35. The Art of Plausible Reasoning

Given a conjecture $P(n)$, depending on an integer parameter n, that has been verified for $1 \leq n \leq M$, how plausible is it?

If it has the form $A(n) \leq B(n)$, then no matter how big M, it would be very stupid to jump to conclusions, as shown in the above examples.

From now we will assume that it can *naturally* be phrased in the form $A(n) = B(n)$. Granted, every assertion $P(n)$, even an inequality like

"$|M(n)| < \sqrt{n}$", is logically equivalent to an *equality* :

$$evalb(P(n)) \equiv true,$$

where $evalb(p)$ is true or false according to whether p is true or false. But of course this is contrived.

The most secure scenario is when *both* A and B are known to belong to a decidable *ansatz* with the strong or weak N_0 property, and it is easy to compute N_0, and it so happened that $M \geq N_0$. Then we immediately have a *rigorous* proof.

Next in line is when A and B both belong to an ansatz with the $N_0(p)$ property and $M \geq N_0(.999999)$ or whatever.

Next in line, as far as plausibility goes, is when there is a strong *heuristic* evidence, inspired by analogy and past experience, that both A and B belong to a *known* ansatz with an N_0 property, M is fairly large, and both A and B are not too complicated.

After that, in the certainty pecking-order, are cases where you have no ansatz in mind to which A and B may possibly belong, but you can *feel it in your bones* that there is a *yet to be discovered ansatz* that would have the N_0 property, and M is fairly large and A and B are not too complicated.

Finally, if the conjecture is so far-out or artificial, or A and B are so different, so that you have no reason to hope that there is a yet-to-be-discovered ansatz that would 'trivialize' $A = B$, and M is not that big, then I wouldn't even make a conjecture.

Also keep in mind the above remarks of finding the right generalization from a one-parameter identity to a multiple-parameter one, that not only can add plausibility to our conjecture, by verifying it for infinitely many cases, but often also facilitates a formal proof.

23.36. Don't Get Hung-Up on the N_0-Approach

In my eyes, an N_0 proof is the *most elegant*. It is also the most fun, since it defies that old and corny platitude, we mathematicians grew up with, that

"**checking finitely many cases, no matter how many, does not constitute a proof**".

But often the N_0 is way too big, and it may not be the most efficient way to prove identities. For example, for the identity

$$(n^{10000000} - 1)(n^{10000000} + 1) = n^{20000000} - 1,$$

it would be stupid to verify it for $1 \leq n \leq 20000001$. Just use the "usual" algorithm for multiplying polynomials.

After all the N_0 approach is just *one* algorithm for proving identities within a given ansatz, and not necessarily always the most efficient one.

23.37. The Wilf-Zeilberger Algorithmic Proof Theory

A less trivial example of an ansatz that has the N_0 property, but using it is usually not feasible, is the WZ algorithmic proof theory, that can prove any conjectured identity of the form

$$\sum_{k=0}^{n} F(n,k) = \sum_{k=0}^{n} G(n,k),$$

whenever $F(n,k)$ and $G(n,k)$ are products of binomial coefficients. By general nonsense we know that the sequence

$$a(n) := \sum_{k=0}^{n} F(n,k) - \sum_{k=0}^{n} G(n,k),$$

is holonomic, i.e. satisfies a homogeneous linear recurrence equation with polynomial coefficients:

$$\sum_{i=0}^{L} p_i(n) a(n+i) \equiv 0,$$

for some non-negative integer L and some polynomials $p_0(n), \ldots, p_L(n)$. It is fairly easy to find relatively small *a priori* upper bounds for L, without actually finding the recurrence. If we knew beforehand that the leading coefficient, $p_L(n)$, has no positive integer zeros, then we could immediately deduce that $a(n)$ is identically 0 once it vanishes for $0 \leq n < L$. Lily Yen, in a 1993 Ph.D. thesis, written under the direction of Herb Wilf, found *a priori* bounds for the largest positive integer root of $p_L(n) = 0$, but they were enormous. It is possible that another approach could bring it down, but why bother? Yen's thesis was interesting *theoretically*, since it showed that WZ theory has the (weak) N_0 property, but as far as actually proving specific identities, it is much more efficient to use the Zeilberger algorithm to actually manufacture the recurrence, and then just look at $p_L(n)$ and convince ourselves that it has no positive integer roots, and if it does find them.

23.38. What Is Mathematical Knowledge?; Reliablism

The standard definition of *knowledge* (see, e.g., Kwame Anthony Appiah's excellent introduction to contemporary philosophy, "**Thinking it Through**") is:

"**justified true belief**".

The problem is then "how justified is justified". In science one is willing to take ample empirical evidence as sufficient justification, but in mathematics, traditionally, one insisted on a formal rigorous proof, proved by human means, since a "proof by computer is only a physical experiment", and "you can't trust a computer", since programs have so many bugs.

Appiah talks about a movement in contemporary epistemology, co-pioneered by my Rutgers colleague Alvin Goldman, called **reliablism** (see also the Wiki entry), that modifies the definition of knowledge to be **true belief justified reliably**. The problem then is to introduce reliability standards.

I strongly believe that very soon most of serious mathematics will be computer-generated, and all of it computer-assisted, so we do need to develop quality-control to maximize the chances that the computer-generated proofs are indeed valid.

One way to maximize reliability is to adopt what I call the **method of overlapping steps**.

Suppose that you have to devise an algorithm to do $S(n), n = 0, 1, 2, \ldots$, and you want to do it for as large n as possible. You should first write the **most naive program**, easy to write, and easy to check. Then let the computer output $S(0), S(1), \ldots, S(L_0)$, with L_0 rather small.

Unfortunately, the naive approach can't go very far. So you write a more sophisticated program, good for $n \leq L_1$, and compare its output with the output of the previous program for $n \leq L_0$. Then you write yet another, even more sophisticated program, valid for $n \leq L_2$ and check it against the previous ones, and so on and so forth.

It can also help if you have two entirely different approaches to tackle the same problem, and if the outputs match, then it is a great indication that they are *both* indeed correct.

There are hardly any isolated facts. As already noticed by Quine, all

knowledge, in particular science and mathematics, consist of intricate *webs*. In the case of computer-generated mathematics, if all your programs are working together without contradiction, this fact simultaneously testifies that they are *all* OK. It is a little like the way computer scientists generate random bits at a fraction of the normal cost by using **expanders**.

23.39. How Necessary is Necessary and How Contingent Is Contingent

Many of the traditional philosophical dichotomies like analytic/synthetic, *a priori/a posteriori*, induction/deduction, and especially necessary/ contingent collapse once we realize that the **mathematical** universe is the **same** as the **physical** universe, and that our **unique** universe is **finite**. Also that everything is **computation**.

I will only dwell on the necessary/contingent dichotomy.

According to traditional thinking, the fact that the speed of light is constant is *contingent*, while the fact that the 100^{th} (decimal) digit of π is 9 is **necessary**. Nonsense. They are both necessary and both contingent. As Greg Chaitin said so beautifully about the digits of Ω, "they are true for no particular reason". But, even if you don't believe in Ω, lots of mathematical facts are, in some sense, contingent, and lots of efforts goes into explaining identities of the form $A = B$ by trying to *explain* them.

Alas, paraphrasing Greg, if the "explanation" is longer than the explanandum, then it is not much of an explanation.

So, the statement

"Amongst any eleven consecutive digits of π, two must be the same"

is much more necessary than the statement

"the 100^{th} digit of π is 9" ,

since the former is a special case of a **universal** result called the **pigeon-hole principle**, with **two** parameters m and n:

$P(m, n)$: If $m > n$ and m pigeons much be placed in n pigeon-holes, then at least two pigeons must be pigeon-hole-mates.

So a numeric result $a = b$, with a and b both numbers (in other words,

they depend on zero parameters), is contingent, even if you have a formal proof. It is less contingent if it is a special case of $A(n) = B(n)$ with $n = n_0$. It is even less contingent if it is a special case of $A(m, n) = B(m, n)$ with $m = m_0, n = n_0$, and so on.

23.40. Depth vs. Elegance

The **depth** of a mathematical result is the smallest amount of computer-time it takes to prove it (within our ansatz). The **elegance** of a statement is how short is its statements.

Paul Erdös believed that God has a book with elegant (i.e. short) proofs of all theorems. I hope that he is wrong. Theorems with short proofs are **shallow**, and my favorite results are short statements that require long proofs.

23.41. Towards a New Kind of Mathematical Aesthetics

Truth is Beauty and Beauty is Truth, or so goes the Keatsian cliche. If Beauty is *elegance*, *symmetry*, and *shortness*, then Beauty is just *trivial* Truth. But if you care about *deep truth*, then you have to give up on the traditional standards of beauty.

23.42. Why Is the Computer-Generated Proof of the Four Color Theorem so Beautiful in My Eyes?

Because the *idea* of the proof can be encapsulated in one short phrase:

There exists an unavoidable set of reducible configurations.

The rest are just details on how to teach the computer to construct such a set, and verify that it is indeed what we want.

In this case, a major open problem was reduced to finding **one** object, that can, and indeed was, searched for, and found, by computer. That **one**, specific, object **certified** that the statement of the Four Color Theorem was indeed true.

A proof of an identity in WZ theory also consists in displaying **one**, finite, object, the WZ certificate, that certifies its correctness. Thereby, apparently, proving "infinitely many cases". Of course, these 'infinitely' many numerical facts are just trivial consequences of just **one** symbolic fact.

23.43. Towards an Ansatz Based Mathematics and Meta-Mathematics

All thinking requires *logic*, but the *informal* logic of normal mathematical discourse is good enough. The reductionist attempt of logicism and formalism to reduce mathematics, at least in principle, to formal logic was unfortunate, even for human-generated mathematics, but especially for computer-generated mathematics. I believe that the logic-based approach that predominates **automatic theorem proving** is not entirely satisfactory.

Also the "abstract nonsense", structuralist, approach, as preached by Bourbaki, was not quite the right approach for humans, and is definitely not suited for computers. Hopefully, the future will bring some synthesis, but, at present, one should try to base mathematical research on *ansatzes*. Major breakthroughs will come not by solving specific open problems, and not even devising new human theories, but by finding new and powerful ansatzes where the open problems can be embedded. It is much more efficient to solve geometry problems using **algebra**, by using analytic geometry, rather than by **logic**, using synthetic geometry.

The traditional dichotomy between **numerical**, empirical, facts, and general, **theoretical** results, is only illusionary. In the eyes of God, $2+2=4$ is just as interesting as Fermat's Last Theorem. It is true that "$2+2=4$" has zero free parameters, while FLT has four $[P(a,b,c,n) := a^n+b^n-c^n \neq 0$ (if $n>2, abc \neq 0)]$, but this is a *quantitative* difference not a *qualitative* one.

Traditionally $(n+1)(n-1)=n^2-1$ is a "theorem", true for infinitely many n, while $3\cdot 5 = 4^2-1$ is just one fact. However, viewed symbolically, they are both facts, the former with one parameter, and the latter with zero parameters.

To prove that 15 is not prime, all you have to do is come up with a factorization: $3\cdot 5 = 15$. For large numbers, this is considered a difficult computational problem, but "conceptually" it is trivial, or so the conventional wisdom says.

To prove that

$$\sum_{k=-n}^{n} (-1)^k \binom{2n}{n+k}^3 = \frac{(3n)!}{n!^3},$$

requires a proof, since this is a general statement, valid for all n, but thanks to WZ theory, there is just **one** object, a certain rational function $R(n,k)$, that certifies it. That certificate can be obtained empirically and algorithmically. So the 'proof' is just one object, like the pair $(3,5)$ in the case of the 'theorem' that 15 is composite.

If desired, it is always possible to convert a 'certificate proof' to a formal logic proof, but this is very artificial, and unnecessary.

Let's conclude this manifesto with:

Mathematicians and meta-mathematicians of the world unite, you have nothing to lose but your logic chains! Let's work together to develop an ansatz-based mathematics and meta-mathematics.

Reminiscences

Chapter 24

In the Company of Giants

Andreea S. Calude

Department of Applied Language Studies and Linguistics, The University of Auckland, Auckland, New Zealand, a.calude@auckland.ac.nz

As the overcrowded bus huddles into the hustle and bustle of Chinatown, after what feels more like a small lifetime than a four-hour journey, you could be almost forgiven for thinking you are in Shanghai, but the signs assure me it is indeed New York that I am finding myself in. Ten minutes later, sitting in a Japanese café and sipping bubble Tapioca tea, I am acting just like one of the locals, well, almost ... aside from my wide-eyed, mesmerized expression.

So here I start my exploration of New York, the Big Apple, the city of opportunities, the land of the yellow cabs, the finance hub, the temple of the 9-11 pain, the house of fashion and art and theatre and music halls. I decide to scrap my NY guide book, and just walk around, absorbing the city that way. I am choosing the skyscrapers as guides instead.

I am due to spend three days in the company of these stirring skyscrapers and while my first visit to the Big Apple makes my heart skip a beat in itself, I am really here for the chance to spend a weekend with a different kind of giant: a self-taught mathematical prodigy who started on his quest of exploring the limits of mathematics at the age of only fifteen, and who discovered the famous Omega number: Greg Chaitin. Featured in countless popular science magazines, *New Scientist, Scientific American, Pour La Science*, Chaitin published his ninth book in September this year. Entitled *Meta Math!* , the book is printed by Random House—not an academic press, but rather the publisher who gave us J. K. Rowling's *Harry Potter*, Dan Brown's *The Da Vinci Code* and Mark Haddon's *The Curious Incident of the Dog in the Night-Time. Meta Math!* is written for anyone with

an interest in mathematical ideas and a curiosity for what it is like to be a mathematician. Chaitin's excitement at his latest creation is unmistakable. Laid out on the small table by the window is the final version of *Meta Math!* with the very last comments and corrections. He shows me the cover, the font, the pagination and the review comments which are to appear on the dust-jacket: "What do you think of them? Do you like them?". Chaitin is passionate about bringing mathematics into the living-rooms of everyday people. He believes that there is a kind of beauty that truly perfect proofs can present to the eye of the mathematician, but it need not be the professional mathematician alone. In his view, you and I can also appreciate it, given the chance.

But if you imagine a mathematical genius to be a quiet, nerdy guy who sits in the corner playing with his wooden abacus, you could not be more wrong in Chaitin's case. As you walk into his house, you realize that "my house, my castle" is a complete understatement for Chaitin, his house is his *world*. It brims with various kinds of art, ranging from colourful Indian prints, to tribal sculptures and statues (including a Maori one!), tasteful female nude paintings and photographs (I spot my favourite among them: Doisneau's *Kiss by the Hotel de Ville*) and art pieces by Birdle, Enhörning and Yeenize. One of the bookcases is filled with Bollywood DVDs, Chaitin's favourite collection. They co-habit in the same room with Rebecca Goldstein's latest book on *Incompleteness* and Hardy's classic *A Mathematician's Apology*. Mathematics, art and beauty are three of Greg's ardent passions and he is the kind of person who will grab anything and everything that reminds him of those things and immediately surround himself by them. His strong personality reigns in every corner of his house.

While browsing Chaitin's varied collections of books, art and DVDs, I am taken back to the feeling of Soho and the shopping district earlier in the day. The streets line up unbrushed and chaotic, pulling you into their tiny chic boutiques, tempting you with various items: tribal and ethnic looking clothes and jewellery, art new and antique, books, CDs, you name it, it's all there. It seems that New York is the 'capital' of the world, and not just a corner of America.

"Grapefruit juice, mineral water and espresso coffee", Chaitin enumerates apologetically the contents of his fridge. He explains that he always eats out, never cooks. So he takes me to an old-fashioned local diner close to his house in Yorktown Heights (an hour outside New York City), where the IBM hub for whom he works is located. As we drive there, I comment

on the abundance of American flags displayed at what feels to be each and every single house we pass. "Of course", he tells me, "every town around here had people who never came back that day". Ground Zero is now fenced off from the public, while the excavating and planning is taking place. As I stood there this morning with my camera, trying to preserve a snapshot of what most people are desperately trying to forget, it occurred to me that even here, in the land of giants and skyscrapers, time and hearts can be made to stop. Sometimes forever.

It turns out that the couch I am about to sleep on tonight is none other than that where Stephen Wolfram himself has also slept in while visiting Chaitin. Wolfram is the author of the celebrated *A New Kind of Science*, the book which challenged existing views of the universe and science as a whole. The Oxford educated physicist is the man behind *Mathematica*, a world-leading software used to solve computer science and mathematical problems, which he moulded in a tool for a series of experiments leading to his credo that small programs can generate enormous complexity – the main theme of his book. I look at the couch inquisitively, expecting it to come alive and speak to me.

Later that Friday night, I ask Chaitin about his views on life, society and the world in general, as we have one last drink. "The problem is that men and women don't like each other anymore" he tells me disheartened. He feels that everything has become a second-hand routine, a kind of sport, a gym membership; people no longer find time to enjoy each other's company and enjoy life, they do not take pleasure in living and loving anymore. Chaitin goes on to say that in spite of the vastness of our universe, there is no place left for the individual; people in general, no longer matter. "I am interested in small countries, like New Zealand, for instance" he adds with noticeable spring in his voice. "It is small countries, and not empires that I am interested in" he adds. There, the individual still matters, voices can still be heard and they do not fade away so easily, ignored, dried up, beaten.

Listening to all this takes me back to the sight of the Empire State Building. It is a colossus, a clump of concrete shooting up into the sky. I feel so small in comparison, a midget, and as I am trying in vain to convince my camera to capture it all in one single shot, the coldness of the building breathes a chilling air onto my skin and suddenly I feel alone. Where did the crowds of bustling people in Chinatown go? Where is everyone? I notice that for the first time, there is no one around. It's just this iceberg

and me. Standing side by side, sizing each other up.

The next day, I am taken to the Hudson River Valley and the mountains which Chaitin is so fond of climbing. Tramping is another one of his many passions. It presents him with another kind of beauty: the beauty of nature. The conversation comes back to mathematics and his work. I ask him how he reacts to the usual criticisms and attacks that someone so famous, working on groundbreaking theories that are bound to disturb a few hairs is sure to endure. "Badly" he offers without a speck of indecision. It seems that everyone who matters is hated somewhere, sometime. It's inescapable. But it is better to be hated than ignored. So when everything goes wrong, there is always the Hudson River Valley and there is always another mountain to climb. Maybe Benjamin Franklin was slightly off with regard to certainty in the world: it is not just taxes and death that are certain.

It's already Sunday and I have one last chance to wonder through the streets of New York. As I am saying my goodbyes to Chaitin at the corner of Central Park, I get distracted momentarily by the street sign "5th Avenue", and realize that there is one last giant I need to visit: The Statue of Liberty. Making my way back to Manhattan and the Financial District, I steal another look at the Empire State Building and smile to myself: *it's been indeed a weekend in the company of many different giants.*

Chapter 25

Gregory Chaitin: Mathematician, Philosopher, and Friend

John Casti

Wissenschaftzentrum Wien, IIASA, The Kenos Circle, Vienna, Austria;
Castiwien@cs.com

In 1990, I was working on the manuscript of my book *Searching for Certainty* (Morrow, New York, 1991), a volume addressing the degree to which the science of today can effectively predict and/or explain various real-world phenomena like weather and climate, the outbreak of warfare and the movement of stock market prices. As part of this story, I wanted to include a chapter on the prediction/explanation of mathematical theorems, in order to open up a discussion of the philosophy of mathematics, especially what we mean by mathematical "truth" and what really constitutes a "proof". Of special concern to me at the time was the question of the *complexity* of a mathematical result, since it seemed clear that whatever limits might be in place for how much science could tell us about the world should be greatly affected by the complexity of the phenomena under consideration. In short, are there phenomena that are simply "too complex" for the human mind to grasp? In particular, are there theorems/mathematical truths that are too complex for our axiomatic systems to actually prove or disprove. This naturally raises the question of what do you mean by the "complexity" of some observed event—including a mathematical proposition. Enter Greg Chaitin.

As part of my background research for the book, I had run across Greg's papers on information theory and algorithmic complexity (later published by World Scientific in his 1992 book, *Information-Theoretic Incompleteness*). Since the whole issue of how much "juice" could you get out of a set of axioms was totally tied up with the notion of incompleteness a la Gödel, Greg's recasting of Gödel's results into the language of computing seemed to be just what I needed to address the 'How much complexity is too much

complexity?' question. So I sent an email to him and asked his view of the matter. Not only did Greg reply (which many name-brand scientists did not when I wrote them about other sections of the book), he generously sent me a voluminous set of papers, references, and ideas about how to frame and explain many of the ideas that ultimately appeared in that chapter. Thus was set in motion an intellectual and personal friendship that is now seventeen-years old and counting.

In this seventeen years, Greg and I have met dozens of times in almost all the world's time zones. And at each one of those meetings I've come away with some bit of knowledge or snippet of information that has caused me to see the world just a bit differently than before. Let me give a rather eclectic account of just a few of those occasions.

Limits to Scientific Knowledge (Santa Fe, NM 1992): In the spring of 1992, Joseph Traub and I organized a two-day workshop at the Santa Fe Institute on the theme, "Limits to Scientific Knowledge" under the sponsorship of the Alfred P. Sloan Foundation. Among the many luminaries at this meeting were biologist Robert Rosen, computer theorist Rolf Landauer, chaologist Otto Rössler, economist Brian Arthur, and Sloan Foundation President Ralph Gomory. But the person who contributed the most to the discussion was, not surprisingly, Greg Chaitin! His booming voice was heard regularly during the intense discussions, commenting on various thoughts and presentations that were floating around the meeting room like the seeds of pollen floating in the desert air of Santa Fe in those days. Greg was a dynamic force that gave both substance and direction to that meeting, and which ultimately led to a follow-up workshop on the same theme in a venue about as far removed from Santa Fe as one can get and still remain on the same planet.

Limits to Scientific Knowledge (Abisko, Sweden 1995): The village of Abisko is located many kilometers north of the Arctic Circle, near the border between Norway and Sweden. For several years, the Swedish Council for Planning and Coordination of Research sponsored an annual meeting organized by Anders Karlqvist and myself on themes residing at the boundary between the natural sciences, philosophy and the humanities. In 1995, Anders and I chose the theme of limits to scientific knowledge, in order to capitalize on the intense, but too short, discussion of these matters in Santa Fe two years earlier. In the Swedish environment, we had a full week of such discussions in very intimate surroundings, as the venue for the meeting was a research station of the Royal Swedish Academy of Sciences, in which all

the participants lived, ate and, in general, spent most of each day together.

Some of the participants in this meeting were the same as in Santa Fe—Greg, Bob Rosen, Joe Traub, Piet Hut, and myself—but several new faces also appeared, including physicist Jim Hartle, biologist Harold Morowitz, and astrophysicist John Barrow. The proceedings of these discussions were published under the title *Boundaries and Barriers: On the Limits to Scientific Knowledge* (Addison-Wesley, Reading, MA, 1996), so I won't go into them here.

What I most remember about this meeting is a conversation I had with Greg Chaitin during a walk one afternoon in the breath-taking surroundings of the research station. We were discussing the question of the complexity of a mathematical theorem, and what one could possibly expect to get out of a given set of consistent axioms. Greg had long before proved that there must be theorems of arbitrarily great complexity, using the notion of algorithmic complexity as the measure, which was already a major extension of Gödel's incompleteness result. But he then went on to state that subject to some technical conditions, it's basically the case that the complexity of the set of axioms sets an upper bound to the complexity of any theorem you can prove within that axiomatic framework. In short, you can't get more out than what you put in.

While it seems self-evident in retrospect, I had never really considered the world of mathematical truths from this perspective before. This result not only makes Gödel's results on the limitations of axiomatic systems much more precise, the philosophical implication is enormous: mathematics is now both limited by the axiomatic system you employ, as well as unlimited by the opportunity to cleverly introduce more axioms to create bigger (i.e., more complex) systems that enable us to prove more complex theorems. But no matter how complex the axiomatic system may be, no single system will ever enable us to "get it all". This is about as direct a statement on the limits to knowledge as one will ever get.

The Infinite (Vienna, Austria, sometime later): During a visit to Vienna, Greg had dinner one evening with myself and his Viennese host, Karl Svozil, in the restaurant *Ofenloch* in the old center of the city. At one point in the conversation, I posed the question: What would a world be like that had no Gödel's Theorem? Of course, this was a provocative question, whose answer rests upon what you believe about the notion of *infinity*, since in a totally finite world, where conceptually infinite objects like the number π

or the square root of 2, do not really exist, then there can be no such results like those of Gödel or Turing. These types of limitative results depend in an essential way upon at least the potentially infinite, if not the actually realized version. So what I was aiming at with this question was really to enquire as to the "reality" of mathematical objects, a long and venerable area of concern in the philosophy of mathematics.

After some deliberation on the question, happily lubricated by some of *Ofenloch*'s fine selection of wines and designer beers, Greg made a remark that I remember to this day. He turned to me and said, "John, you have to remember that the infinite is very powerful!" Very powerful, indeed! So powerful, in fact, that our entire view of the world would be turned upside down if it could ever be proved that the universe is, in fact, strictly finite.

These are but a few of the many interactions I've had with Greg that have impacted my intellectual and personal life in a major way. In the end, the questions we have discussed and debated have all been much more philosophical than mathematical. And though the world, at-large regards Greg as a "mathematician", when you read his autobiographical volume *Meta Math!* (Pantheon, New York, 2005) it is impossible not to be struck by the deeply philosophical—and emotional—content of Greg's work.

So I salute you, Greg, on this the occasion of your 60^{th} birthday. I'm happy to have the privilege of knowing you and to have learned so much from our interactions. May you have at least sixty years more to reach many more minds with your wisdom, intelligence, and never-ending set of novel and imaginative ideas.

Œuvre

Chapter 26

Algorithmic Information Theory: Some Recollections

Gregory Chaitin

IBM Research, Yorktown Heights, USA; chaitin@us.ibm.com

Introduction

AIT is a theory that uses the idea of the computer, particularly the size of computer programs, to study the limits of knowledge, in other words, what we can know, and how. This theory can be traced back to Leibniz in 1686, and it features a place in pure mathematics where there is absolutely no structure, none at all, namely the bits of the halting probability Ω.

There are related bodies of work by other people going in other directions, but in my case the emphasis is on using the idea of algorithmic complexity to obtain incompleteness results. I became interested in this as a teenager and have worked on it ever since.

Let me tell you that story. History is extremely complicated, with many different points of view. What will make my account simple is the unity of purpose imposed on a field that is a personal creation, that has a central spine, that pulls a single thread. What did it feel like to do that? In fact, it's not something I did. It's as if the ideas wanted to be expressed through me.

It is an overwhelming experience to feel possessed by promising new ideas. This happened to me as a teenager, and I have spent the rest of my life trying to develop the ideas that flooded my mind then. These ideas were deep enough to merit 45 years of effort, and I feel that more work is still needed. There are many connections with crucial concepts in other fields: physics, biology, philosophy, theology, artificial intelligence... Let me try to remember what happened to me... The history of a person's life, that's just gossip. But the history of a person's ideas, that is real, that is

important, that is where you can see creativity at work. That is where you can see new ideas springing into being.

AIT in a Nutshell

Gödel discovered incompleteness in 1931 using a version of the liar paradox, "This statement is unprovable." I was fascinated by Gödel's work. I devoured Nagel and Newman, *Gödel's Proof*, when it was published in 1958.

I was also fascinated by computers, and by the computer as a mathematical concept. In 1936 Turing derived incompleteness from uncomputability. My work follows in Turing's footsteps, not Gödel's, but adds the idea of looking at the size of computer programs.

For example, let's call a program Q "elegant" if no program written in the same language that is smaller than Q produces the same output. Can we prove that individual programs are elegant? In general, no. Any given formal axiomatic system can only enable us to show that finitely many programs are elegant.

It's easy to see that this must be so. Just consider a program P that calculates the output of the first provably elegant program that is larger than P. P runs through all the possible proofs in the formal axiomatic system until it finds the first proof that an individual program Q larger than P is elegant, and then P runs Q and returns Q's output as its (P's) output.

If you assume that only true theorems can be proved in your formal axiomatic system, then P is too small to be able to produce the same output as Q. If P actually succeeds in finding the program Q, then we have a contradiction. Therefore Q is never found, which means that no program that is bigger than P can be proven to be elegant.

So how big is P? Well, it must include a big subroutine for running through all the possible proofs of the formal axiomatic system. The rest of P, the main program, is rather small; P is mostly that big subroutine. That's the key thing, to focus on the number of bits in that subroutine.

So let's define the algorithmic complexity of a formal axiomatic system to be the size in bits of the smallest program for running through all the proofs and producing all the theorems. Then we can state what we just proved like this: You can't prove that a program is elegant if its size is substantially larger than the algorithmic complexity of the formal axiomatic

system that you are using.

Instead of saying "a formal axiomatic system of algorithmic complexity N," I'll just say "N bits of axioms." So if you have N bits of axioms, then no program larger than $N + c$ bits in size can be proven to be elegant. That's the result we just proved.

A more sophisticated example is the number I call Ω, which is the halting probability of a computer running a program produced one bit at a time by repeatedly tossing a coin. Because it is a probability, this number has to be between zero and one. Imagine writing it out in binary:

$$\Omega = .011100\ldots$$

These bits are peculiar, they are irreducible mathematical information. This means that a formal axiomatic system with N bits of axioms can enable you to determine at most $N + c$ bits of Ω. Essentially the only way to determine bits of Ω is to add that information directly to your axioms. Even though Ω is a single well-defined real number (once you fix the programming language), its bits have no structure, no pattern, none at all, they are irredundant, irreducible mathematical information.

In other words, the bits of Ω are mathematical facts that are true for no reason, no reason simpler than themselves.

So that's the basic idea, and those are my two favorite results, but the devil is in the details. You can spend your life on those details, and I did.

Chaitin Research Timeline

- **1947**: Born in Chicago, child of Argentine immigrants. Family moves to New York.
- **1956**: Nagel and Newman's article on "Gödel's proof" is published in *Scientific American*. Article contains a photo by Arnold Newman of Gödel sitting in front of an empty blackboard at the Princeton Institute for Advanced Study.
- **1958**: Nagel and Newman's book *Gödel's Proof* is published by New York University Press.
- **1959**: Following directions in the *Scientific American* "Amateur Scientist" department, I build a Van de Graaff generator for high-voltage static electricity.

- **1962**: First year at Bronx High School of Science. While answering an essay question on the entrance exam for the Columbia University Science Honors Program for bright high school students, I get the idea of defining randomness using program-size complexity.

  The essay question is what do you conclude if you find a pin on the moon.[1] My answer is that this means that somebody must have visited before you, because a pin is not natural, it is artificial, the product of intelligence. And, I remark, what this means is that there is a small program to calculate it, to create it. That's how we can tell that the pin has structure and is artificial. And, contrariwise, something natural would not have a description that can be compressed into a small program, because it was not designed.

  And then, as a throw-away remark, I state that a random thing is one that cannot be compressed into a smaller program. More precisely, I am speaking about a digital description of an object, not about the object itself. In other words, in 1962 I give the following

  - **Definition of Randomness R1**: A random finite binary string is one that cannot be compressed into a program smaller than itself, that is, that is not the unique output of a program without any input, a program whose size in bits is smaller than the size in bits of its output.

  However I quickly forget about this definition, because I am having so much fun learning how to write, debug and run computer programs in the Science Honors Program. And I am given the run of the math stacks at Columbia University and can hold in my hands and study the collected works of Euler and other priceless volumes.

- **1963**: Shannon and McCarthy, *Automata Studies*, Princeton University Press, 1956, contains E. F. Moore's paper "Gedanken-experiments on sequential machines."[2] Following Moore, I write a program for identifying a finite-state black box by putting in inputs and looking at the outputs. My experiments suggest this is easier to do than Moore anticipated. I prove this to be the case in a note "An improvement on a theorem of E. F. Moore" (*IEEE Transactions on Electronic Computers*, 1965), my first publication.

[1] This was before the first lunar landing.

[2] I became aware of Shannon and McCarthy, perhaps the first book on the theory of computation, because it was reviewed in *Scientific American*.

- **1964**: Summer vacation between high school and college, I try to find an infinite set with no subset that is easier to generate than the entire set. By easier I mean faster or simpler; at this point I am simultaneously exploring run-time complexity and program-size complexity. The work goes well, but is not published until 1969 in the *ACM Journal* as "On the simplicity and speed of programs for computing infinite sets of natural numbers."

  Also that summer, I get the first incompleteness result that I will publish, **UB1**, an upper bound on the provable lower bounds on run-time complexity in any given formal axiomatic system. This is published in 1970 in a Rio de Janiero Pontifícia Universidade Católica research report, and only there.[3]

  Another discovery that summer, **UB2**, is that one can diagonalize over the output of all programs that provably calculate total functions $f\colon N \to N$ to obtain a faster growing computable total function $F\colon N \to N$. That is to say, given any formal axiomatic system, one can construct a computer program from it that calculates a total function $f\colon N \to N$, but the fact that this program calculates a total function $f\colon N \to N$ cannot be proved within the formal axiomatic system, because f goes to infinity too quickly. "Calculates a total function $f\colon N \to N$" merely means that every time we give the program f a natural number n as input, it eventually outputs a single natural number $f(n)$ and then halts.

  The result UB1 is actually a corollary of UB2, since all lower bounds on run-time complexity are computable total functions.

  Now one would say that the proof of UB2 is an instance of Cantor diagonalization, but in my opinion it's really closer to Paul du Bois-Reymond's theorem on orders of infinity. His theorem is that for any scale of rates of growth, any infinite list of functions that go to infinity faster and faster, for example

$$f_0(n) = 2^n,$$
$$f_1(n) = 2^{2^n},$$
$$f_2(n) = 2^{2^{2^n}} \ \ldots,$$

[3]While writing up that report in Rio, I realize I can also obtain an upper bound on the provable lower bounds on program-size complexity.

there is another function

$$f_\omega(n) = \max_{k \le n} f_k(n)$$

that goes to infinity even more quickly. As far as I know, Paul du Bois-Reymond's work was independent of Cantor's.

Note the Cantor ordinal number ω as a subscript. We can then form

$$f_{\omega+1}(n) = 2^{f_\omega(n)},$$
$$f_{\omega+2}(n) = 2^{f_{\omega+1}(n)},$$
$$f_{\omega+3}(n) = 2^{f_{\omega+2}(n)} \ldots$$

and then

$$f_{2\omega}(n) = \max_{k \le n} f_{\omega+k}(n).$$

Continuing in this manner, we get to

$$f_{3\omega} \ldots f_{4\omega} \ldots f_{\omega^2} \ldots f_{\omega^3} \ldots f_{\omega^\omega} \ldots f_{\omega^{\omega^\omega}} \ldots$$

and onwards and upwards, incredibly fast-growing functions all.

I had read about Paul du Bois-Reymond's work in a monograph by G. H. Hardy called *Orders of Infinity*.[4]

My point of view changes between 1964 and 1965. In my 1964 work, only **infinite** computations are considered. In 1965, on the contrary, only **finite** computations are considered, computations that produce **a single output**. Furthermore, my interest shifts from run-time complexity to program-size complexity.

- **1965**: During the spring term of my first year at City College, CUNY, I simultaneously study three books: von Neumann and Morgenstern, *Theory of Games and Economic Behavior*, Shannon and Weaver, *The Mathematical Theory of Communication*, and Turing's 1936 paper "On computable numbers..." in the anthology Davis, *The Undecidable*. The 1962 definition of randomness (R1) comes back to me; as I will now explain, all three books play a vital role. It all comes together as I am reading a discussion in von Neumann and Morgenstern of the game of matching pennies, for which their theory says that you should toss a coin.[5]

[4] I learned the calculus from Hardy's *A Course of Pure Mathematics*, and also enjoyed *A Mathematician's Apology* and Hardy and Wright, *An Introduction to the Theory of Numbers*.

[5] One of the players is trying to match the other player's choice of head or tails.

In a footnote they remark that the theory of games seems to require a quantum-mechanical world in which God plays dice. Not really, I say to myself. Another logical possibility would be that the theory of games tells you to use a random sequence of choices, but you cannot compute this sequence of choices from the theory.

You see, in the game of matching pennies, if a theory can tell you exactly what to do, you can predict what your opponent will do and beat him. The solution to the paradox is either that the theory asks you to use physical randomness, which is unpredictable, or that you have a theory that says you must make mathematically random or unstructured choices, and the contradiction is avoided because these are in fact uncomputable (a notion taken from Turing).

The third leg of the stool comes from reading Shannon, who defines a message to be random or have maximum entropy if it cannot be compressed, if it cannot be encoded more compactly. Obviously the most general possible **decoder** would be a universal Turing machine, a general-purpose computer.

With my 1962 definition (R1), I have managed to connect game theory, information theory and computability theory. Now all I have to do is work out the details.

Now it's the summer vacation between my first and second year at City College, and I attempt to carry out the plan. At the beginning of the summer, the road forward seems blocked, but I keep trying. Later in the summer, ideas begin to flood into my mind. I write a single paper that is the size of a small book. At the request of the editors, I later divide it in two, and delete much material to save space.

This paper "On the length of programs for computing finite binary sequences"—part one is published in 1966 and part two is published in 1969, both in the *ACM Journal*—presents **three** different theories of program-size complexity (and embryonic versions of ideas that I would explore for years):

- **Complexity Theory (A)**: Counting the number of states in a normal Turing machine with a fixed number of tape symbols. I call this Turing machine state-complexity.
- **Complexity Theory (B)**: The same as theory (A), but now there's a fixed upper bound on the size of transfers—jumps,

branches—between states. You can only jump nearby. I call this bounded-transfer Turing machine state-complexity.

– **Complexity Theory (C)**: Counting the number of bits in a binary program, a bit string. The program starts with a **self-delimiting** prefix, indicating which computing machine to simulate, followed by the binary program for that machine. That's how we get what's called **a universal machine**.[6]

Let's define the complexity of a bit string to be the size of the smallest program that computes it.

In each case, theory (A), (B) or (C), I show that most n-bit strings have complexity close to the maximum possible, and I determine asymptotic formulas for the maximum possible complexity of an n-bit string. These maximum or near maximum complexity strings are defined to be random. To show that this is reasonable, I prove, for example, that these strings are "normal" in Borel's sense. This means that all possible blocks of bits of the same size occur in such strings approximately the same number of times, an equi-distribution property.

I start with theory (A) because that seems the most straightforward thing to do. The idea in theory (A) is to eliminate all the redundancy in a real programming language. Then I switch to theory (B), in which I don't eliminate the redundancy, I live with it. The proofs are prettier; more subtle, not so heavy-handed. However, in theories (A) and (B) I cannot figure out how to show that a **small** amount of structure in an n-bit string will force its complexity to dip below the maximum possible complexity for n-bit strings.

To solve this, I switch from Turing machines to binary programs and theory (C). Theory (C) solves the problem, but feels too easy, like stealing candy from a baby. For example, in theory (C) it is trivial to show that most n-bit strings have close to the maximum possible complexity. And this maximum possible complexity is precisely $n + 1$, not an asymptotic estimate as in theories (A) and (B).[7]

[6] I call theory (C) "blank-endmarker" program-size complexity, to distinguish it from "self-delimiting" program-size complexity, theory (D) below.

[7] To get (max complexity n-bit string) $= n + 1$, theory (C) has to be a bit more complicated.

– **Complexity Theory (C2)**: If the first bit of the program is a 0, then output the rest of the program as is and halt. If the first bit of the program is a 1, process the rest of the program as in theory (C).

By the way, in theories (A) and (B), randomness definition (R1) does not apply, because the size of programs is measured in states, not bits. It is necessary to use a slightly different definition of randomness:

– **Definition of Randomness R2**: A random n-bit string is one that has maximum or near maximum complexity. In other words, an n-bit string is random if its complexity is approximately equal to the maximum complexity of any n-bit string.

In theory (C), (R1) works fine (but so does (R2), which is more general).

Theory (C) is essentially the same as the one independently proposed by Kolmogorov at about the same time (1965).[8]

However, I am dissatisfied with theory (C); the absence of **subadditivity** disturbs me. What is subadditivity? The usual definition is that a function f is subadditive if $f(x + y) \leq f(x) + f(y)$. I mean something slightly different. Subadditivity holds if the complexity of computing two objects together (also known as their **joint complexity**) is bounded by the sum of their individual complexities.[9] In other words, subadditivity means that you can combine subroutines by concatenating them, without having to add information to indicate where the first subroutine ends and the second one begins. This makes it easy to construct big programs. Complexity is subadditive in theories (A) and (B), but not in theory (C).

Last but not least, "On the length of programs for computing finite binary sequences" contains what I would now call a Berry paradox proof that program-size complexity is uncomputable. This seed was to grow into my 1970 work on incompleteness, where I refer to the Berry paradox explicitly for the first time.

- **1966**: Awarded by City College the Belden Mathematical Prize and the Gitelson Medal "for the pursuit of truth." Family moves back to Buenos Aires.
- **1967**: I join IBM Argentina, working as a computer programmer.

(C2) is the version of theory (C) given in "On the length of programs for computing finite binary sequences: statistical considerations" (*ACM Journal*, 1969), the second of the two papers put together from my 1965 randomness manuscript.

[8]Solomonoff was the first person to publish the idea of program-size complexity—in fact, (C)—but he did not propose a definition of randomness.

[9]In the case of joint complexity the computer has **two** outputs, or outputs **a pair** of objects, whatever you prefer.

- **1969**: Stimulated by von Neumann's posthumous *Theory of Self-Reproducing Automata*, I work on a mathematical definition of life using program-size complexity. This is published in Spanish in Buenos Aires, and the next year (1970) in English in the *ACM SICACT News*. This is the first of what on the whole I regard as an unsuccessful series of papers on theoretical biology.[10]

- **1970**: I visit Brazil and inspired by this tropical land, I realize that one can get powerful incompleteness results using program-size arguments. In fact, one can place **upper** bounds on the provable **lower** bounds on run-time **and** program-size complexity in a formal axiomatic system. And this provides a way to measure the power of that formal axiomatic system.

  This first information-theoretic incompleteness result is immediately published in a Rio de Janiero Pontifícia Universidade Católica research report and also as an *AMS Notices* abstract, and comes out the next year (1971) as a note in the *ACM SIGACT News*.

  I obtain a *LISP 1.5 Programmer's Manual* in Brazil and start writing LISP interpreters and inventing LISP dialects.[11]

- **1971**: I write a longer paper on incompleteness, "Information-theoretic limitations of formal systems," which is presented at the Courant Institute Computational Complexity Symposium in New York City in October 1971. A key idea in this paper is to measure the complexity of a formal axiomatic system by the size in bits of the program that generates all of the theorems by systematically running through the tree of all possible proofs.

- **1973**: I complete a greatly expanded version of "Information-theoretic limitations of formal systems." The expanded version appears in the *ACM Journal* in 1974. A less technical paper on the same subject, "Information-theoretic computational complexity," is presented at the IEEE International Symposium on Information Theory, in Ashkelon, Israel, June 1973, and is published in 1974 as an invited paper in the *IEEE Transactions on Information Theory*.[12]

[10] The latest one is "Speculations on biology, information and complexity" (*EATCS Bulletin*, February 2007).

[11] Around 1973, I give courses on LISP and on computability and metamathematics at the University of Buenos Aires.

[12] In 1974 I send a copy of this paper to Kurt Gödel, leading to a pleasant, short phone

- **1974**: I am invited to visit the IBM Watson Lab in Yorktown Heights for a few months. The visit goes well, with a number of major breakthroughs. I realize what to do to theory (C) to restore subadditivity, and discover the halting probability Ω.
  - **Complexity Theory (D)**: Counting the number of bits in a self-delimiting binary program, a bit string with the property that you can tell where it ends by reading it bit by bit without ever reading a blank endmarker. Now a program starts with a self-delimiting prefix as before, but the program to be simulated that follows the prefix must **also** be self-delimiting. So the idea is that the **whole** program must now have the same property the **prefix** already had in theory (C).

  (D) is the mature theory, I believe. I immediately put this out as an IBM research report. I present this paper at the opening plenary session of the IEEE International Symposium on Information Theory in Notre Dame, Indiana, in October 1974. It is published in the *ACM Journal* in 1975 as "A theory of program size formally identical to information theory."

  There are three key ideas in this paper: self-delimiting programs, a new definition of relative complexity, and the idea of getting program-size results **indirectly** from probabilistic, measure-theoretic arguments involving the probability $P(x)$ that a program will calculate x. I call this the algorithmic probability of x.[13] Summing $P(x)$ over all possible outputs x yields the halting probability Ω:

$$\Omega = \sum_x P(x).$$[14]

conversation with Gödel and an appointment to meet him at the Princeton Institute for Advanced Study, an appointment that Gödel cancels at the last minute.

[13] Solomonoff tried to define $P(x)$ but could not get $P(x)$ to converge since he wasn't working with self-delimiting programs.

[14] This definition is a bit abstract. Here are two other ways of defining an Ω number. As a sum over programs p:

$$\Omega' = \sum_{p \text{ halts}} 2^{-|p|}.$$

As a sum over all positive integers n:

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

Here $|p|$ is the size in bits of the program p, and $H(n)$ is the size in bits of the smallest program for calculating the positive integer n.

And a key theorem

$$(*) \quad H(x) = -\log_2 P(x) + O(1)$$

permits us to translate complexities into probabilities and vice versa. Here the complexity $H(x)$ of x is the size in bits of the smallest program for calculating x, and the $O(1)$ indicates that the difference between the two sides of the equation is bounded.

Incidentally, $(*)$ implies that there are few minimum or near-minimum size programs for calculating something, few minimal descriptions. That is, an elegant program for calculating something is essentially unique.[15]

Where did the halting probability Ω come from? How did I come up with it? Well, already in part two of my first paper on randomness, "On the length of programs for computing finite binary sequences: statistical considerations" (*ACM Journal*, 1969), I use a Heine-Borel-style algorithm to exhibit a specific example of a random infinite sequence of bits. I think it is important to come up with specific examples.

The halting probability Ω is a natural example of a random infinite sequence of bits. Besides providing a connection with the work of Turing, Ω makes randomness more concrete and more believable.

Furthermore, once you have a natural example of randomness, you immediately get an incompleteness theorem from it, as I point out in "Gödel's theorem and information" (*International Journal of Theoretical Physics*, 1982). My work in 1987 on Ω and its diophantine equation makes this fully explicit. I had been aware of this opportunity for getting a dramatic incompleteness result for some time.[16]

$(*)$ is nice but it is only part of the story. The true reward for changing from theory (C) to (D) is this spectacular result:

$$(**) \quad H(x, y) = H(x) + H(y|x) + O(1).$$

In words, (the joint complexity of two objects) is equal to the sum of (the absolute complexity of the first object) plus (the relative complexity of the second object given the first object).

[15] For more on this, see my book *Exploring Randomness* (2001). I had proven this result in theory (C) in 1972, but that proof wasn't published until 1976, in "Information-theoretic characterizations of recursive infinite strings" in *Theoretical Computer Science*.

[16] See the end of the introductory section of "Information-theoretic limitations of formal systems" (*ACM Journal*, 1974).

To get (∗∗), self-delimiting programs aren't enough, you also need the right definition of **relative complexity**.[17] I had used relative complexity in my big 1965 randomness manuscript, but had eliminated it to save space. In my 1975 *ACM Journal* paper I take up relative complexity again, but define $H(x|y)$, the complexity of x given y, to be the size in bits of the smallest self-delimiting program for calculating x if we are given for free, not y directly, but a minimum-size self-delimiting program for y.

And (∗∗) implies that the extent to which computing two things together is cheaper than computing them separately, also known as the mutual information

$$H(x:y) \equiv H(x) + H(y) - H(x,y),$$

is essentially the same, within $O(1)$, of the extent to which knowing x helps us to know y

$$H(y) - H(y|x),$$

and this in turn is essentially the same, within $O(1)$, of the extent to which knowing y helps us to know x

$$H(x) - H(x|y).$$

This is so pretty that I decide never to use theory (C) again. For (D) doesn't just restore subadditivity to (C), it reveals an elegant new landscape with sharp results instead of messy error terms. From now on, theory (D) only.

- **1975**: My first *Scientific American* article, "Randomness and mathematical proof," appears. I move back to New York and join the IBM Watson Lab.

  In the paper "Algorithmic entropy of sets" (*Computers & Mathematics with Applications*, 1976, written at the end of 1975), I attempt to extend the self-delimiting approach to programs for generating infinite sets of output. Much remains to be done.[18]

  This topic is important, because I think of a formal axiomatic system as a computation that produces theorems. My measure of the complexity

[17] Levin claims to have published theory (D) first. However he missed this vital part of theory (D).

[18] If this interests you, please see the discussion of infinite computations in the last chapter of *Exploring Randomness* (2001).

of a formal axiomatic system is therefore the size in bits of the smallest **self-delimiting** program for generating the infinite set of theorems.

- **1976–1985**: I concentrate on software and hardware engineering for IBM's RISC (Reduced Instruction Set Computer) project.

  Even though I spend most of my time on this engineering project, in 1982 I publish "Gödel's theorem and information" in the *International Journal of Theoretical Physics*.[19] This is later included with my 1974 *IEEE Transactions on Information Theory* paper in the influential anthology Tymoczko, *New Directions in the Philosophy of Mathematics*, Princeton University Press, 1998.

  My 1985 publication "An APL2 gallery of mathematical physics: a course outline" gives computational working models of physical phenomena to be inserted between chapters of Einstein and Infeld, *The Evolution of Physics*. This APL2 physics simulation software is extremely concise.[20]

- **1986**: My RISC engineering work stops because of an invitation by Cambridge University Press to write the first volume in their series Cambridge Tracts in Theoretical Computer Science. I start working on the book, intending merely to collect previous results, but then the flow of new ideas resumes.

  Some of these new ideas are presented in the paper "Incompleteness theorems for random reals" (1987). This paper contains a proof of the basic result that an N-bit formal axiomatic system cannot enable you to determine more than $N+c$ bits of Ω, bits that may be scattered and do not have to be together at the beginning of Ω. This is the measure-theoretic proof that I give in the Cambridge book. After writing this paper, work on the book begins in earnest.

  Using work by Jones and Matijasevic on Hilbert's 10th problem, I calculate a 200-page diophantine equation for Ω.[21] This equation has thousands of unknowns and a parameter k, and has finitely or infinitely many whole-number solutions depending on whether the kth bit of Ω is, respectively, a 0 or a 1. To get this equation, I convert a register

[19] At roughly the same time I fulfill a childhood dream by building my own telescope: I join a telescope-making club and grind a 6 inch f/8 mirror for a Newtonian reflector in a basement workshop at the Hayden Planetarium of the Museum of Natural History.

[20] Besides learning the physics and some numerical analysis, I wanted to get a feel for the algorithmic complexity of the laws of physics.

[21] It's actually what's called an **exponential** diophantine equation.

machine program for a LISP interpreter into a diophantine equation, and then I plug into that equation a LISP program for computing lower bounds on Ω.

- **1987**: Cambridge University Press publishes *Algorithmic Information Theory*, which explains how to obtain the diophantine equation for Ω. This book also contains a result about random infinite binary sequences. I show that four definitions of this concept are equivalent: two constructive measure-theoretic definitions due to Martin-Löf and to Solovay, and two definitions of my own using program-size complexity. *Algorithmic Information Theory* is my first publication in which LISP appears.

  Simultaneously, World Scientific publishes a collection of my papers, *Information, Randomness and Incompleteness*.

- **1988**: I write about the diophantine equation for Ω in my second *Scientific American* article, "Randomness in arithmetic" (1988), and then in *New Scientist* (1990), and after that in *La Recherche* (1991).

- **1991**: I give a lecture on "Randomness in arithmetic" in the room where Gödel taught at the University of Vienna.

- **1992**: In the 1992 paper on "Information-theoretic incompleteness," I publish a program-size proof of the theorem that you cannot determine the bits of Ω. More precisely, an N-bit theory—a formal axiomatic system with complexity N—can permit you to determine at most $N + c$ bits of Ω. This program-size proof is better than the measure-theoretic proof in the 1987 Cambridge book, and is the proof that I use in the 1998 book, *The Limits of Mathematics*.

  I also publish four papers on LISP program-size complexity:

  – **Complexity Theory (L)**: Counting the number of characters a LISP S-expression (that's the program) must have, to have a determined value (that's the output).

  In these four papers several different dialects of LISP are studied, and appropriate versions of the halting probability Ω_{LISP} are invented for each of them, together with the corresponding incompleteness theorems. The techniques developed in my complexity theories (A) and (B) are put to good use here.

  These LISP Ω numbers may not be as random as the fully random Ω number in theory (D), but, like the so-called random infinite sequences

in my original 1965 randomness paper, they come close. For example, they are Borel normal for blocks of all sizes in any base.

These 1992 papers are immediately included in my second World Scientific volume, *Information-Theoretic Incompleteness* (1992).

I lecture at a meeting on reductionism at Cambridge University. The transcript of that lecture, "Randomness in arithmetic and the decline and fall of reductionism in pure mathematics," appears later in Cornwell, *Nature's Imagination*, Oxford University Press, 1995.

- **1995**: I discover how to convert theory (D) into a theory about the size of real programs, programs that you can actually run on a computer (universal Turing machine) that I simulate using a special version of LISP.
  - **Complexity Theory (E)**: Counting the number of bits in a self-delimiting binary program, a bit string with the property that you can tell where it ends by reading it bit by bit without reading a blank endmarker. The program starts with a self-delimiting prefix, indicating which computing machine to simulate, followed by the self-delimiting binary program for that machine, as in theory (D). But in theory (E), (the prefix indicating the machine to simulate) is a LISP S-expression, a program written in a high-level functional programming language, that's converted to a bit string. (E) isn't a new theory, it's a special case of (D) selected because the prefix indicating the machine to simulate is encoded in a particularly convenient manner.

  Now, 30 years after starting to work on program-size complexity, I can finally run programs and measure their size.

  Several years are needed to complete this concrete version of AIT and to present it in my three Springer-Verlag volumes, *The Limits of Mathematics* (1998), *The Unknowable* (1999), and *Exploring Randomness* (2001). These books come with LISP software and a LISP interpreter.

  Honorary doctorate, University of Maine.

- **2000**: Since this year, visiting professor, Computer Science Department, University of Auckland, New Zealand.

- **2002**: Honorary professor, University of Buenos Aires.

  Springer-Verlag publishes *Conversations with a Mathematician*, a collection of some of my lecture transcripts and interviews.

I'm invited to present a paper at a philosophy congress in Bonn, Germany, in September. For this purpose, I begin to study philosophy, particularly Leibniz's work on complexity, which I am led to by a hint in a book by Hermann Weyl.

My paper appears two years later (2004) in a proceedings volume published by the Academy Press of the Berlin Academy that was founded by Leibniz. It is reprinted as the second appendix in my book *Meta Math!* (2005).

- **2003**: Lecture notes *From Philosophy to Program Size* published in Estonia, based on a course I give there, winter 2003.
- **2004**: Corresponding member, *Académie Internationale de Philosophie des Sciences.*

  Write *Meta Math!*, a high-level popularization of AIT, published the following year by Pantheon Books (2005). This book is not just an explanation of previous work; it presents a *système du monde.*

  Meta Math! follows Émile Borel in questioning the existence of the bulk of the real numbers, a train of thought further developed in my paper "How real are real numbers?" (*International Journal of Bifurcation and Chaos*, 2006).[22]
- **2005**: I summarize the *système du monde* of *Meta Math!* in the paper "Epistemology as information theory: From Leibniz to Ω" (***Collapse**: Journal of Philosophical Research and Development*, 2006).

  Honorary president of the scientific committee of the Valparaíso Complex Systems Institute in Chile. Member of the scientific advisory panel of FQXi, devoted to Foundational Questions in Physics & Cosmology.
- **2006**: The centenary of Gödel's birth. I publish my third *Scientific American* article, on "The limits of reason," celebrating Leibniz, whom Gödel also admired. This article is translated and published in about a dozen other languages. I summarize my thoughts on incompleteness in an Enriques lecture at the University of Milan, "The halting probability Ω: Irreducible complexity in pure mathematics" (*Milan Journal of Mathematics,* 2007).

[22] Vladimir Tasić pointed out to me that Borel has a know-it-all real number—a 1927 version of the Ω number. My paper on the ontological status of real numbers is dedicated to Borel's memory.

A collection of some of my philosophy papers is published in Turin. Full member, *Académie Internationale de Philosophie des Sciences.*

- **2007**: World Scientific publishes a more complete collection of my philosophy papers. I establish a connection between the bits of Ω and the word problem for semigroups in my paper "An algebraic characterization of the halting probability" (*Fundamenta Informaticae,* 2007). 60th birthday.

To paraphrase Einstein, this timeline will have fulfilled its purpose if it shows how the efforts of a lifetime hang together, and why they lead to certain definite expectations.[23] In particular, I think it would be fruitful to explore the following topics.

Challenges for the Future

- On the technical side, many questions remain regarding the program-size complexity and the algorithmic probability of computing infinite sets of objects.

More difficult challenges:

- To develop a model of mathematics that is biological, that is, that evolves and develops, that's dynamic, not static. Perhaps a time-dependent formal axiomatic system?
- To understand creativity in mathematics—where do new ideas come from?—and also in biology—how do new, much more complicated organisms develop? Perhaps a life-as-evolving-software model has some merit?
- Ω = concentrated creativity? Each bit of Ω = one bit of creativity? Can human intellectual progress be measured by the number of bits of Ω that we know, or are currently capable of knowing, as a function of time?
- Is the universe discrete or continuous? Can physical systems contain an infinite or only a finite number of bits?
- Make a model world in which you can prove life must develop with high probability. It doesn't matter if this world isn't exactly like ours; how can that be important?
- Is the world built out of information, not matter? Is it built out of thought? Is matter just an epiphenomenon, that is, secondary, not

[23] See the final sentence of Einstein's *Autobiographical Notes.*

primary? And what are thoughts?

Selected Publications by Chaitin

Non-Technical Books

- *Meta Math!*, Pantheon Books, New York, 2005 (hardcover); Vintage, New York, 2006 (paperback).
- U.K. edition: *Meta Maths*, Atlantic Books, London, 2006.[24]
- *Conversations with a Mathematician*, Springer-Verlag, London, 2002.[25]

Technical Books

Lecture Notes

- *From Philosophy to Program Size*, Institute of Cybernetics, Tallinn, 2003.

Monographs

- *Algorithmic Information Theory*, Cambridge University Press, 1987 (hardcover), 2004 (paperback).
- *The Limits of Mathematics*, Springer-Verlag, Singapore, 1998; reprinted by Springer-Verlag, London, 2002.[26]
- *The Unknowable*, Springer-Verlag, Singapore, 1999.[27]
- *Exploring Randomness*, Springer-Verlag, London, 2001.

Collections of Papers

- *Information, Randomness and Incompleteness*, World Scientific, Singapore, 1987, 2nd ed., Singapore, 1990.
- *Information-Theoretic Incompleteness*, World Scientific, Singapore, 1992.
- *Teoria algoritmica della complessità*, Giappichelli Editore, Turin, 2006.
- *Thinking about Gödel and Turing*, World Scientific, Singapore, 2007.

[24] Also published in Greek.
[25] Also published in Japanese and Portuguese.
[26] Also published in Japanese.
[27] Also published in Japanese.

Celebration

Chapter 27

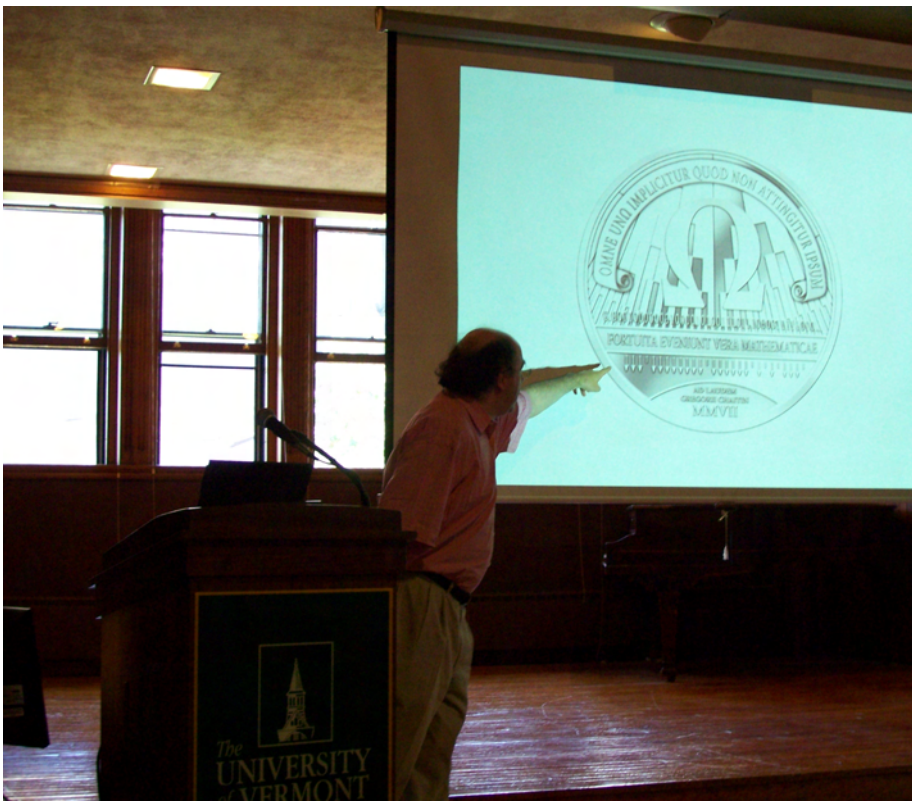
Chaitin Celebration at the NKS2007 Conference

On 15 July 2007 the *New Kind of Science 2007 Conference* Conference (Burlington)[1] hosted a special Chaitin celebration. It included a session dedicated to the book "Randomness & Complexity, from Leibniz to Chaitin", a special lunch honouring Gregory Chaitin, and a panel discussion on *Randomness and Complexity* with Cristian Calude, John Casti, Gregory Chaitin, Paul Davies, Karl Svozil, and Stephen Wolfram.

The special session included the following presentations:

- Cristian Calude, *Proving and Programming*
- John Casti, *Greg Chaitin: Twenty Years of Personal and Intellectual Friendship*
- Karl Svozil, *The Randomness Information Paradox: Recovering Information in Complex Systems*
- Paul Davies, *The Implications of a Cosmological Information Bound for Complexity, Quantum Information and the Nature of Physical Law*
- Gordana Dodig-Crnkovic, *Where Do New Ideas Come From? How Do They Emerge? Epistemology as Computation (Information Processing)*
- Ugo Pagallo, *Chaitin's Thin Line in the Sand. Information, Algorithms, and the Role of Ignorance in Social Complex Networks*
- Hector Zenil, *On the Algorithmic Complexity for Short Sequences*
- Gregory Chaitin, *On the Principle of Sufficient Reason*

[1] `http://www.wolframscience.com/conference/2007/program.html`.



Stephen Wolfram presenting the Omega Medallion (photo by Cristian Calude)

Special session contributors. From left to right: Hector Zenil, Stephen Wolfram, Paul Davies, Ugo Pagallo, Gregory Chaitin, Cristian Calude, Karl Svozil, Gordana Dodig-Crnkovic, and John Casti (photo by Sally McCay)

"Randomness and Complexity" panellists. From left to right: Paul Davies, Gregory Chaitin, Stephen Wolfram, John Casti, Karl Svozil, and Cristian Calude (photo by Jeff Grote)

Stephen Wolfram, Gregory Chaitin, and the Omega cake (photo by Jeff Grote)

Omega cake approximation (photo by Karl Svozil)