CDMTCS
Research
Report
Series

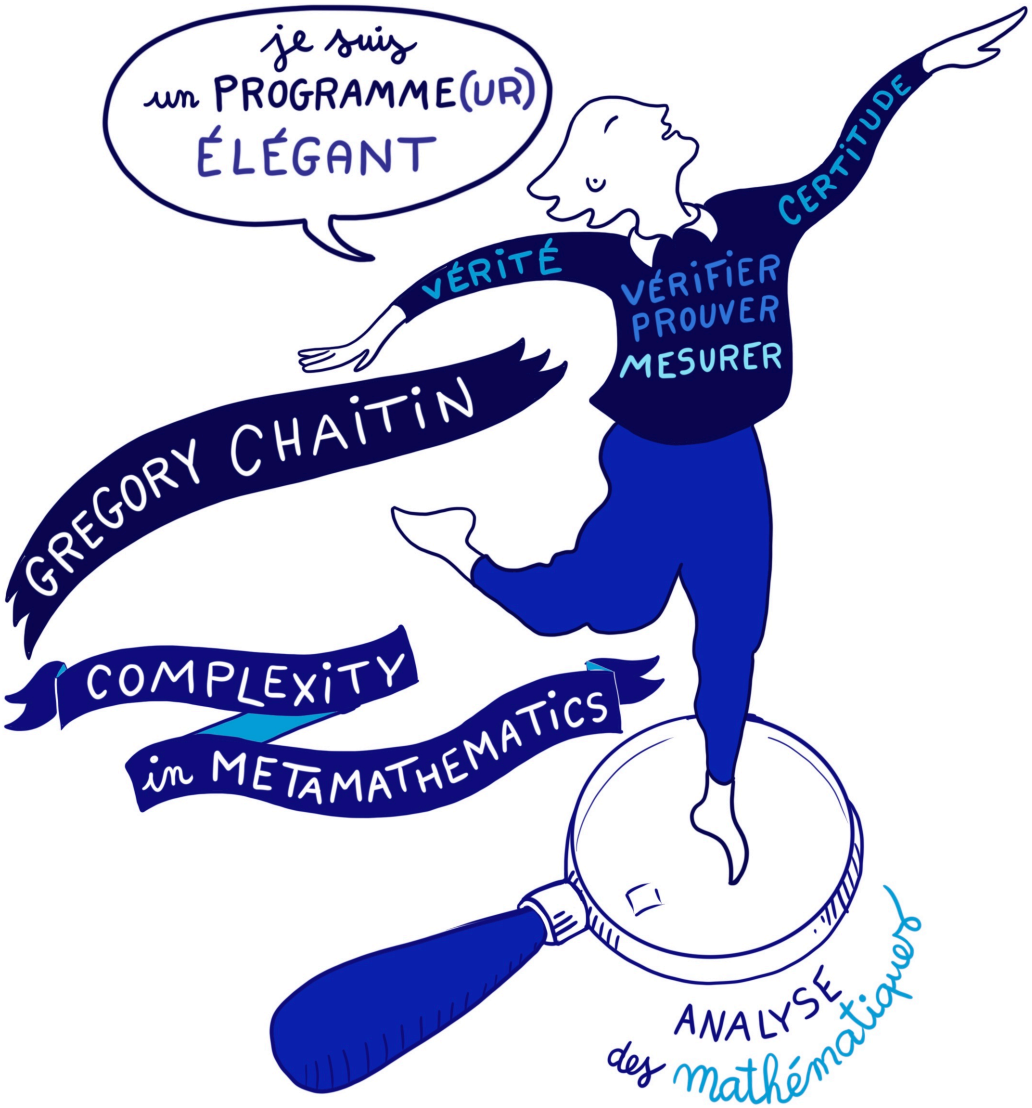
Construire l'Univers à Partir de l'Information et du Calcul

Gregory Chaitin

University of Buenos Aires, Argentina

CDMTCS-570
April 2022

Centre for Discrete Mathematics and
Theoretical Computer Science


je suis un PROGRAMME(UR) ÉLÉGANT
VÉRITÉ
VÉRIFIER PROUVER MESURER
CERTITUDE
GREGORY CHAITIN
COMPLEXITY in METAMATHEMATICS
ANALYSE des mathématiques

Construire l'univers à partir de l'information et du calcul

Building the World from Information & Computation

G. J. Chaitin
https://uba.academia.edu/GregoryChaitin

March 29, 2023

Contents

Advertisement

Weaving together mathematics, epistemology and autobiography, this book begins with a tribute to Leibniz and a guide to Greg's favorite books, then presents Greg's three favorite lecture transcripts, and last but not least tells the story of Greg's unconventional life and research trajectory and ponders the mystery of creativity. Seventy pages, mostly in English, with a foreword and a brief introduction in French intended for a Moroccan audience. The material in French was kindly provided by Réda Benkirane, a professor at UM6P, and author of the book *La Complexité, vertiges et promesses*. Cover illustration by Sophie Lenormand based on Greg's talk at UM6P in Chapter 3.

Avant-propos par Réda Benkirane

Mathématicien et chercheur émérite au Centre de recherche IBM à New York, Gregory Chaitin est, tout simplement, l'homme qui a spécifié, à la fin du XXe siècle, les limites conceptuelles des mathématiques. Sa plus grande découverte est un nombre inconnaissable. C'est le nombre aléatoire oméga (Ω)*, au sujet duquel les mathématiques restent totalement muettes... Et pourtant, une équation longue de 200 pages a été nécessaire pour le générer !*

Mathématicien prodige, Gregory Chaitin, citoyen américain d'origine argentine, situe dès le début de sa carrière sa démarche au carrefour de la théorie de l'information et de l'informatique. Sa spécialité est la théorie de la complexité algorithmique, qu'il a conceptualisée en parallèle avec l'éminent mathématicien russe Kolmogorov. Et les titres de ces livres évoquent bien l'ampleur d'une recherche qu'il a poursuit tout au long de son itinéraire scientifique : il a tout d'abord approfondi les relations entre « information, hasard et incomplétude », puis a tenté d'esquisser de la manière la plus raisonnable qui soit « les limites des mathématiques », avant de réfléchir finalement à la nature de « l'inconnnaissable ».

Gregory Chaitin, qui appartient à cette génération de mathématiciens pour lesquelles l'informatique a été un nouveau moyen d'exploration de la profondeur mathématique, a développé une réflexion épistémologique sur le devenir des mathématiques à l'ère de l'ordinateur. Il travaille aux limites actuelles des mathématiques — limites de calculabilité et de logique — et, fasciné par l'univers des nombres, recherche la manière la plus concise de les générer et de les compresser.

Précisons toutefois que l'étrangeté des resultats auxquels a abouti Chaitin n'est pas due à l'utilisation d'une causalité circulaire, d'un raisonnement interne, ou à aucun autre « exotisme » mathématique de type auto-référentiel, mais au contraire à la découverte de l'immensité, épaisse et insondable, des données mathématiques qui débordent tout cadre théorique. L'ordinateur ici ne sert pas à simuler le réel mais à y creuser jusqu'à atteindre des profondeurs qui l'homme ne pourrait, de toute façon, jamais atteindre par d'autres moyens.

Gregory Chaitin se situe dans la lignée intellectuelle de deux grands logiciens du XXe siècle, Kurt Gödel et Alan Turing qui, tout deux, avec leurs théorèmes sur l'incomplétude et sur l'incalculabilité, ont placé la science mathématique face à ces limites. Dans certains domaines et dans certains cas, la certitude peut ne pas exister ou n'être d'aucune utilité pour la pensée mathématique. Ce constat doit-il pour autant reléguer la mathesis *à une point de vue comme une autre sur le monde ? Pour Gregory Chaitin, rien n'est moins sûr !*

Chapter 1

Complexité, logique et hasard. Entretien avec Gregory Chaitin par Réda Benkirane (2002)

> « Vous pouvez déjà clairement voir le commencement d'un nouveau genre de mathématiques, un genre plus compliqué qui, d'une certaine manière, ressemble à la biologie. » — Gregory Chaitin, *The Unknowable*

— *Vous êtes connu pour avoir développé la théorie algorithmique de l'information. Les mathématiques semble vous passionner depuis toujours, si l'on considère que vous avez écrit vos premières contributions scientifiques, par exemple, sur la notion de compression, alor que vous étiez adolescent...*

— D'une certaine manière, je suis un autodidacte en mathématiques. J'ai eu cette idée de la mesure de la complexité d'un objet mathématique à l'age de quinze ans et j'ai publié mon premier article important sur le subjet à l'âge de dix-neuf ans dans le *Journal of the Association for Computing Machinery*. J'ai eu un parcours assez inhabituel...

— *Vos recherches sur la complexité algorithmique sont très techniques, tournées vers la science informatique, mais les implications de vos travaux sont, elles, d'ordre philosophique et finalement assez abordables en dehors de votre discipline. En fait, vous montrez qu'il existe des vérités mathématiques sans démonstration possible, sans structure ni modèle, des vérités qui n'ont aucune explication. Vous révélez l'existence d'objets mathématiques produits uniquement par l'hasard. Comment réagit la communauté des mathématiciens á vos travaux ?*

Il n'y a pas eu dans le monde des mathématiques beaucoup de réactions par rapport à ma recherche sur le hasard, de même qu'il n'y avait pas eu beaucoup de réactions aux travaux du logicien Kurt Gödel sur l'incomplétude. La plupart des mathématiciens travaillent sur des problèmes relevant de leur domaine technique particulier. Ce travail de *métamathématiques*, propre notament aux recherches de Gödel, de Turing, ou aux miennes, reste en dehors des champs de recherche spécialisés. Le grand publique s'est montré davantage intéressé... Dans la communauté scientifique, les physiciens ont manifesté plus de sympathie pour ce que j'ai fait que les mathématiciens. Cela correspond mieux à leur point de vue sur le monde et à leur façon de penser. La plupart des ouvrages de vulgarisation qui évoquent mes travaux viennent de physiciens — je

pense par example à John Barrow et Paul Davies. Je trouve deux causes à cet intérêt des physiciens. D'abord, je montre que les mathématiques ne conduisent pas à une vérité absolue mais sont quasi empiriques ; je considère qu'elles sont plus proches de la physique que ne veulent le croire les mathématiciens en général. Je ne veut pas dire par là que les deux disciplines scientifiques se confondent, elles sont clairement distinctes, mais mon approche consiste plutôt à mettre en avant les similarités que les différences. En second lieu, j'utilise des idées issues des sciences physiques : l'idée du hasard a été beaucoup plus étudiée par la physique au cours du XXe siècle. Lorsque Einstein dit que « Dieu ne joue pas aux dés », il le fait en reference à la méchanique quantique. D'ailleurs, j'ai été invité à participer à un numéro spécial de *La Recherche* consacré à la théorie du chaos, qui est une théorie physique. La même chose est arrivé en Grande-Bretagne avec la revue *New Scientist*.

L'idée du hasard n'est pas une idée avec laquelle les mathématiciens se sentent à l'aise : ils ne croient pas au hasard, ils ont tendance à penser que les choses sont noires ou blanches, vraies ou fausses, ils considèrent que les choses surviennent uniquement en fonction d'une raison particulière. Quelques mathématciens, toutefois, portent attention à mes idées. Aux États-Unis, John Allen Paulos, qui est connu par ses livres grand public sur les mathématiques, s'y est intéressé — il est intéressé par la philosophie, ce qui est inhabituel pour un mathématicien !

À ma connaissance, il n'y a qu'un philosophe, Thomas Tymoczko, l'auteur de *New Directions in the Philosophy of Mathematics*, qui ait réagit un peu à mes travaux... Mon travail est interdisciplinaire et il est parfois difficile pour des spécialistes de comprendre de quoi je parle !

— *Vous définissez la complexité algorithique d'un objet mathématique en fonction de « la taille du plus petit programme qui le calcule ou produit sa description complète »...*

— Une autre manière de le dire serait de définir la complexité algorithmique en fonction de la théorie la plus concise qui puisse l'expliquer.

— *Vous précisez que « des objets plus simples requièrent des programmes plus courts ». Ce qui est désigné comme un programme informatique — un ensemble d'instructions codées en chaînes de 0 et 1 — est un nombre décrivant un autre nombre mais dans une séquence généralement plus courte. Dans cette perspective, un nombre aléatoire n'a pas de compression possible ; le plus petit programme susceptible de le décrire consiste à écrire entièrement ce nombre. Existe-t-il des définitions mathématiques — plus courtes ou plus longues – du hasard ?*

— Il y a d'autres définitions, en particulier la définition statistique du hasard due au Suédois Martin-Löf. Sa définition s'appuie sur les propriétés statistiques du hasard alors que ma définition est fondée sur la notion de complexité de l'information. Un jolie théorème a prouvé que nos deux théories étaient équivalentes, ce qui est toujours une bonne chose en mathématiques ! Lorsque, pour un même objet, plusieurs définitions proposées se rèvélent équivalentes, cela indique que vous avez réellement touché le coeur du problème, que vous avez identifié les concepts les plus pertinents.

En ce qui concerne la complexité algorithmique et la taille des programmes informatiques, deux autres personnes ont eu l'idée d'y travailler : Solomonoff et Kolmogorov. Solomonoff nà jamais parlé de hasard, le problème central qui l'intéressait était l'intelligence artificielle, il parlait de prédiction et soulevait des questions liées au problème de l'induction scientifique et au principe du rasoir d'Occam (qui affirme que

les meilleures théories sont les plus simples). Kolmogorov et moi-même avons développé, en parallèle et à peu près à la même periode, une réflexion sur la question du hasard. Nous pensions tous deux que cette idée était importante pour définir le manque de structure ou de motif.

— *Concrètement, quelle utilité y a-t-il à connaître la complexité algorithmique d'un objet ? Ce concept pourrait-il servir, par exemple, dans la discipline émergente qu'est la cryptographie, dont l'une des applications est de générer des chiffres aléatoires ?*

— Ma théorie sur les nombres aléatoires est surtout théorique ! Je pense que l'application la plus intéressante est l'épistémologie. Le fait le plus marquant de cette théorie est qu'elle prouve qu'elle est impraticable, qu'elle n'a pas d'application parce que vous ne pourrez jamais calculer le degré de complexité ! Vous pourrez prouver des théorèmes la concernant, mais il sera impossible d'être jamais sûr du degré de complexité d'un objet en particulier. Même si, selon ma définition, la plupart des choses sont aléatoires, vous ne pourrez jamais en être sûr ! C'est ce qui rend ma théorie impraticable, inutile du point de vue des applications scientifiques et technologiques, mais c'est aussi ce qui la rend fascinante philosophiquement parlant.

Beaucoup de travail a été accompli par les théoriciens de l'informatique dans le domaine de la cryptographie, mais je pense que la plupart de ces études recourent à des concepts plus pratiques que les miens, peut-être parce que ces chercheurs sont plus intéressés par le monde réel que moi-même...

— *Doit-on en déduire que vous travaillez surtout sur un plan épistémologique et philosophique ?*

— C'est ce que j'essaie de faire. Mais mon travail est en réalité connecté aux idées fondamentales de la physique. Ma mesure de la complexité algorithmique est très proche de la notion d'entropie développée par Ludwig Boltzmann (1844–1906), fondamentale en physique statistique et en thermodynamique et qui mesure le degré de désordre d'un système physique. Cette notion d'entropie, dès l'époque de Boltzmann, entretenait des liens solides avec la philosophie, par exemple en ce qui concerne la direction, la flèche du temps. Boltzmann s'est suicidé en partie parce que sa théorie très controversée de l'entropie croissante, postulant que les systèmes physiques passent d'un état ordonné à un état désordonné, impliquait une direction du temps, en contradiction flagrante avec la physique newtonienne qui, étrangement, n'avait pas, elle, de direction du temps. Au lieu d'accueillir favorablement la théorie de Boltzmann, qui était plus proche de la vie courante — nous savons tous que tout finit par vieillir et se casser —, certains de ses collègues la rejetèrent parce que, justement, elle semblait contredire la physique newtonienne où, si vous connaissez le présent de l'état d'un système donné, vous pouvez prédire aussi bien son passé que son futur.

Une autre connexion importante est celle avec la biologie, où l'ADN est semblable à un programme d'information biologique servant à construire des organismes, à développer un embryon et à construire un être vivant. Ces notions d'information et de complexité sont beaucoup discutées actuellement parce que les sociétés humaines, notre technologie et les organismes biologiques sont des organisations complexes. Ce que je propose est une théorie de la complexité dans le cas le plus simple possible, celui des mathématiques pures. Ce n'est donc pas une théorie réaliste et pratique, mais elle montre au moins qu'il existe une notion de complexité mathématiquement concevable. Le domaine des mathématiques pures est beaucoup plus simple que le monde réel, le monde de la biologie ou celui de la physique. Ma théorie de la complexité est en fait

plus facile car elle s'applique uniquement à l'épistémologie et non au monde réel. Elle atteste uniquement des limites du raisonnement mathématique, mais j'espère qu'elle suggère que de nouvelles définitions, de nouveaux concepts de complexité émergent au sein de chaque discipline pour être utiles dans le monde réel.

— *Vous avez montré qu'il existe des nombres aléatoires, incompressibles. Vous avez notament découvert un nombre appelé oméga* (Ω)*, qui est la probabilité d'arrête d'un ordinateur universel, et qui ne possède aucune structure. Cela veut dire qu'il n'est pas possible de distinguer les bits de la valeur numérique d'oméga du resultat d'un lancer de pièce de monnaie. Quels enseignements tirez-vous de ce nombre ?*

— C'est un nombre fascinant, une sorte de cauchemar pour la raison pure. Il a une définition mathématique precise, mais il est inconnaissable, incalculable. Donc, d'un point de vue rationnel, oméga est très effrayant. Quelles implications peut-on en tirer ? C'est une question très controversée mais selon moi, oméga montre bien que les mathématiques sont plus proches de la physique que ne le croient les mathématiciens. Oméga est un cas extrème où, pour être en mesure de prouver plus, vous avez besoin de plus d'hypothèses et d'axiomes. En fait, c'est un cas où la seule façon de prouver ce à quoi ce nombre correspond en bits, c'est-à-dire en mode binaire, il faut ajouter ce que vous essayez de prouver en nouveaux axiomes. En d'autres termes, c'est irréductible, le raisonnement mathématique n'est réellement d'aucune aide. Pour des physiciens, cela ne semble pas trop outrageux comme affirmation car, au fur et à mesure que progresse la physique, de nouvelles hypothéses sont ajoutées — comme ça a été le cas par exemple pour les équations de Newton, Maxwell, Einstein, Schrödinger...

Extrait du livre Réda Benkirane, *La Complexité, vertiges et promesses*, Éditions Le Pommier, 2002.

Chapter 2

Building the World from Information & Computation. A Guide to Some of the Relevant Literature (2021)

> "As God cogitates and calculates, so the world is made." — Leibniz

Digital Philosophy

According to Pythagoras: **All is Number, God is a Mathematician**. Modern physics is in fact based on continuous mathematics, differential and partial differential equations, validating Pythagoras' vision. In this essay we shall instead discuss a neo-Pythagorean ontology: **All is Algorithm, God is a Programmer**. In other words, can there be discrete computational models of the physical world? This is sometimes referred to as digital philosophy. There are in fact two books on digital philosophy, both in Italian:

- Ugo Pagallo, *Introduction to Digital Philosophy, from Leibniz to Chaitin*
- Andrea Vaccaro and Giuseppe Longo, *Bit Bang: The Birth of Digital Philosophy*

Let's review the development of digital philosophy. We'll start, as is often the case, with Leibniz.

Leibniz on Information and Computation

> "I have so many ideas that may perhaps be of some use in time if others more penetrating than I go deeply into them some day and join the beauty of their minds to the labor of mine." —G. W. Leibniz (1646–1716)

Good sources of information on the universal genius and visionary Leibniz are

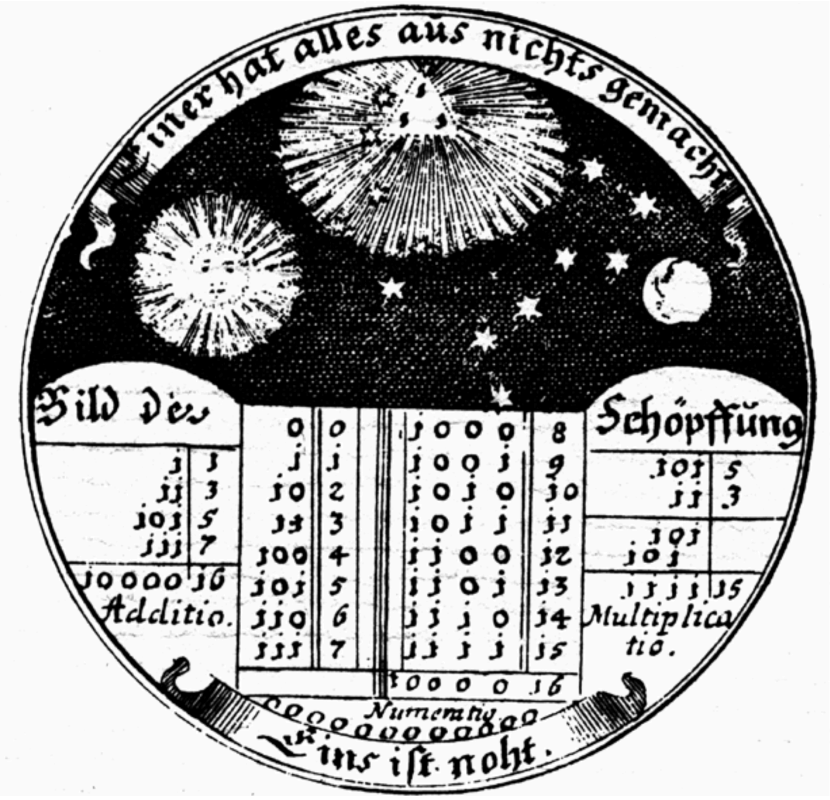
- Eric Temple Bell, *Men of Mathematics*

which has an entire chapter on him, and

- Tobias Dantizig, *Number, The Language of Science*

on the evolution of the number concept, in which references to Leibniz are scattered throughout the text.

Here we draw freely on these two sources. But they do not include this design made by Leibniz for a medal to celebrate his discovery that the entire universe could be built out of 0s and 1s, out of binary information:



A literal translation of the German is: "The one has all from nothing (zero) made. Picture of the Creation." Put more gracefully, "One suffices to derive all out of nothing. Image of the Creation." And below the sun, moon, stars and an image of the Godhead, the divine principle of creation, we see examples of base-two, binary addition and multiplication and the integers from 0 to 16 in both decimal and binary.

This is a breathtaking vision, but one that was not understood by another genius, Laplace, the Newton of France, author of a massive five-volume treatise on celestial mechanics. Says Laplace:

> "Leibniz saw in his binary arithmetic the image of Creation. ... He imagined Unity represented God, and Zero the void; that the Supreme Being drew all beings from the void, just as unity and zero express all numbers in his system of numeration. This conception was so pleasing to Leibniz that he communicated it to the Jesuit, Grimaldi, president of the Chinese tribunal for mathematics, in the hope that this emblem of creation would convert the Emperor of China, who was very fond of the sciences. I mention

> this merely to show how the prejudices of childhood may cloud the vision even of the greatest men!"

This is beautifully written, like his two masterpieces, *Exposition of the System of the World*, and *Philosophical Essay on Probabilities*, but Laplace could not anticipate our contemporary digital world. Leibniz could. He realized that 0 and 1 had the combinatorial potential to represent anything. He not only proposed information as the ontological basis for the universe, he constructed a calculating machine that could multiply. Pascal's earlier machine could only add. These were in fact the artificial intelligence projects of the time. Leibniz displayed his machine in London, before the priority fight over the invention of the calculus soured everything, and was named a foreign member of the Royal Society, which later, under Newton's presidency, viciously attacked Leibniz.

Furthermore, Leibniz realized just how extremely valuable it would be to possess machines that could calculate:

> "And now that we may give final praise to the machine we may say that it will be desirable to all who are engaged in computations which, it is well known, are the managers of financial affairs, the administrators of others' estates, merchants, surveyors, geographers, navigators, astronomers... But limiting ourselves to scientific uses, the old geometric and astronomic tables could be corrected and new ones constructed by the help of which we could measure all kinds of curves and figures... it will pay to extend as far as possible the major Pythagorean tables; the table of squares, cubes, and other powers; and the tables of combinations, variations, and progressions of all kinds... Also the astronomers surely will not have to continue to exercise the patience which is required for computation... For it is unworthy of excellent men to lose hours like slaves in the labor of calculation which could safely be relegated to anyone else if the machine were used."

Here we draw upon another valuable source:

- Martin Davis, *The Universal Computer: The Road from Leibniz to Turing*

And this leads us to Turing and the 20th century.

Universal Machines and Universal Programming Languages

One idea which it seems that Leibniz did not have—but one would have to read all his letters to be sure—is the idea of a universal machine, a contribution of the 20th century:

- Alan Turing, 1936/1937, "On Computable Numbers, with an Application to the *Entscheidungsproblem*" in Martin Davis, *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*

The basic idea in Turing's paper is the software/hardware distinction, and that by varying the software it is possible for a single universal computing machine to simulate the functioning of any other digital information-processing piece of machinery or circuitry. We call these *general-purpose computers*, as opposed to *special-purpose calculating machinery*.

But Leibniz did have the idea of a universal language, the *caracteristica universalis*.

Programming languages are an important new kind of linguistic object. They are much less ambiguous than natural human language, because one has to explain everything to a mere machine. It turns out that most programming languages are in fact universal: any algorithm can be programmed.

Biology

Plasticity in computer technology and in the biosphere is due to the fact that both are based on software. Nature discovered software eons before humanity did.

This beautiful idea appears for the first time in

- John von Neumann, 1949 (lecture), 1951 (paper), "The General and Logical Theory of Automata" in Jeffress, *Cerebral mechanisms in behavior; the Hixon Symposium*

where he is courageous enough to refer to computers as *artificial automata* and to biological organisms as *natural automata*. The fundamental mathematical idea in both cases, according to von Neumann, is the idea of software. Indeed, DNA is a kind of digital software for constructing and running an organism. And measuring the biological complexity of an organism in terms of the size of its DNA, suggests also taking a look at the size of computer programs. In one case the unit of measurement is base pairs, in the other case it is bits of software.

This paper of von Neumann's inspired Sydney Brenner to be one of the creators of molecular biology. He shared an office at Cambridge University for many years with the Crick of Watson and Crick. My wife and I had the privilege of discussing biology with Brenner in Singapore shortly before his death, confined to a wheelchair but as brilliant and incisive as ever.

In an attempt to go a step beyond von Neumann's 1949/1951 formulation, the embryonic field dubbed *metabiology* proposes applying random mutations to computer programs instead of to DNA. The goal is to see what one can prove.

Here is a book on this, a philosophical overview, and an excellent discussion in *Quanta* magazine:

- Gregory Chaitin, *Proving Darwin: Making Biology Mathematical*
- Virginia Chaitin, "Metabiology, Interdisciplinarity and the Human Self-Image" in Wuppuluri, Doria, *Unravelling Complexity: The Life and Work of Gregory Chaitin*
- https://quantamagazine.org/computer-science-and-biology-explore-algorithmic-evolution-20181129/

Algorithmic Information Theory (AIT)

We've mentioned biological complexity, but what about conceptual complexity: the complexity of physics theories, and the complexity of mathematical theories?

The simplicity of physics theories is a topic discussed by many scientists and philosophers of science, for example Ernst Mach and Henri Poincaré. Especially relevant to our present discussion are the following texts, which influenced the present author:

- Karl Popper, *The Logic of Scientific Discovery*
- Hermann Weyl, *Philosophy of Mathematics and Natural Science*
- Hermann Weyl, *The Open World: Three Lectures on the Metaphysical Implications of Science*

Philosopher of science Popper has an entire chapter on simplicity in his book. Mathematician Weyl has brief but penetrating comments on complexity in both his philosophical texts. And he discovered and emphasizes the significance of Leibniz' profound remarks on complexity, remarks in a relatively short text in the Leibniz *nachlass* that was only found amongst his papers well after his death:

- Leibniz, *Discours de métaphysique, Discourse on Metaphysics*

Here is Leibniz' formulation: The empirical data consists of a finite set of points on a piece of graph paper giving the behavior of a physical system as a function of time, the theory is an equation passing through those points, and the complexity is the size of that equation.

And here is how AIT reformulates this: Now everything is 0s and 1s. The empirical data is a finite binary string, the theory is a binary computer program for calculating the data, and the complexity is the number of bits of software in the program for the theory. The size in bits of the best theory, which is the smallest program to calculate the data, is the algorithmic complexity or information content of the data. The smallest program to calculate the data is the best theory for it.

In a similar spirit, the complexity of a formal axiomatic mathematical theory is the size in bits of the smallest program that systematically runs through the tree of all possible proofs finding all the theorems, an interminable computation. This is possible because the influential mathematician Hilbert insisted that a properly formalized axiomatic theory should contain an algorithm for checking syntax and for checking the validity of proofs.

Using this complexity measure for mathematical theories, one can formulate a number of powerful limitative metamathematical theorems in the spirit of Gödel's famous incompleteness theorem, thus shedding a new light on the limitations of formal axiomatic reasoning.

The Language of Creation

So the complexity is the number of bits of software required to program the physics or math theory. But this depends on the programming language that is being used!

Which programming language to use in measuring complexity took a few years to straighten out, work that was done by the present author and a few others. The candidates are the programming languages that permit the most concise programs and that satisfy a few other more technical desiderata, such as that the complexity of a composite object should be bounded by the sum of the individual complexities of its parts. This is referred to as *subadditivity*.

Using terminology from the Middle Ages, and returning to the metaphor of God the Programmer, God's language of creation must be one of these concise, subadditive languages.

All of this, which is rather technical mathematics, is discussed in the following publications:

- Gregory Chaitin, "The Perfect Language," https://inference-review.com/article/the-perfect-language, Chapter 5 in this book.
- Gregory Chaitin, *Algorithmic Information Theory*

and also in essays in these two *festschriften*:

- Cristian Calude, *Randomness and Complexity, from Leibniz to Chaitin*
- Shyam Wuppuluri, Francisco Antonio Doria, *Unravelling Complexity: The Life and Work of Gregory Chaitin*

Quantum Information and Quantum Computation

So much for digital philosophy with classical, deterministic 0/1 bits. But the world is actually probabilistic and quantum mechanical, not classical. So now let's turn to building the world from quantum information and quantum computation instead of classical, ordinary information and computation. This is something that physicists are actually working on right now, as you can see from the title of this recent book:

- Vlatko Vedral, *Decoding Reality: The Universe as Quantum Information*

The microscopic quantum world is very different from the classical macroscopic world. In quantum mechanics one encounters strange things like probability waves that interfere constructively and destructively. And quantum *qubits*, not classical 0/1 bits.

A qubit is a probabilistic superposition of 0 and 1, a quantum mixture, a mixed state rather than a pure state. For example, it might be 90% 0 and 10% 1, or 50% 0 and 50% 1. Furthermore, quantum probabilities have a phase (direction) as well as a magnitude, so they can interfere and even cancel rather than reinforce each other. In other words, adding an additional way that something can happen can make it less likely than before, which never happens in classical (non-quantum) physics.

Quantum computation is a Feynman path integral = sum over all possible time-evolution paths = sum over all histories. In the quantum world you fly from Rio to NY simultaneously using several different airlines, changing planes in all possible places. This is called quantum parallelism. The ability to simultaneously pursue an exponential number of different computational paths on a single computer is what makes quantum computing so powerful.

And the quest to build the world from qubits is now a hot research topic. For example, there are efforts to get spacetime to emerge from the quantum entanglement of qubits. Here is an article in *Quanta* magazine discussing such work by the physicist Erik Verlinde:

- https://quantamagazine.org/erik-verlindes-gravity-minus-dark-matter-20161129/

According to Verlinde, gravity is not a fundamental force. It is an entropic force, like osmosis, that emerges statistically.

Does Quantum Mechanics (QM) Need to be Modified?

Building the world from qubits would be terrific—and would validate Leibniz' vision—but we have been living with the philosophical conundrums of QM for a century. An example is the measurement problem: how does a QM mixed state collapse into a pure (classical) state when a measurement is performed? One proposal is that the consciousness of the observer intervenes, another is that the universe forks, which is called the many-worlds interpretation of QM. No generally accepted solutions have been found. The only way out is to modify or replace QM. Any chance of this happening?

In fact, according to Randell Mills, QM may have been a mistake! There is an entire book about this:

- Brett Holverstott, *Randell Mills and the Search for Hydrino Energy*

Here is his story. Randell Mills studied physics at MIT with Hermann Haus, who used novel solutions of Maxwell's equations for electromagnetism—which is classical physics—to explain the functioning of the free electron laser. Randell Mills took his professor's work on the free electron laser and applied it to the hydrogen atom, obtaining a classical theory with a non-radiating electron, precisely the problem that quantum mechanics was invented to solve! And his theory predicts a novel form of atomic hydrogen that he calls *hydrinos*.

According to Mills, hydrinos are atomic hydrogen below the normal ground state, with the electron closer to the proton than conventional QM allows, because the electron is a hollow sphere around the proton, not a point circling it.

Amazingly enough, there is much laboratory evidence for hydrinos. Please see the experimental data collected at https://brilliantlightpower.com, Randell Mills' private research organization.

Furthermore, Mills believes that hydrinos are the mysterious dark matter. They would then make up most of the substance of the world, since something like 90% of the mass of the universe is estimated to be the invisible dark matter. This is all well and good, but most unfortunately it may call into question building the world from qubits and quantum computation. Fortunately, hot off the press, Stephen Wolfram has a breath-taking new way to build the world from discrete, classical information and computation: rewriting rules that are successively applied to graphs.

From String Theory to Wolfram's Project to Find the Fundamental Theory of Physics

A major problem in contemporary physics is that General Relativity (GR) and QM are incompatible. String theory attempted to solve this problem and was extremely fashionable for a time, but now seems to be stuck, not to mention unable to make testable predictions.

I should tell you about Wolfram's new project to create discrete computational models of physics, work which I consider to be extremely interesting. He has already published a big, fat, beautifully illustrated book about this:

- Stephen Wolfram, *A Project to Find the Fundamental Theory of Physics*

It turns out that in his framework general relativity and quantum mechanics emerge as different aspects of essentially the same idea. This means that his models provide an alternative to string theory, which was created to solve the problem of making QM and GR compatible.

He explains this in the section labeled "GR and QM are the same idea!" on pages 55 through 57 of the first chapter of his book, and concludes triumphantly thusly:

> "In physical space we have Einstein's equations—the core of general relativity. And in branchial space (or, more accurately, multi-way space) we have Feynman's path integral—the core of modern quantum mechanics. And in the context of our models they're just different facets of the same idea. It's an amazing unification that I have to say I didn't see coming; it's something that just emerged as an inevitable consequence of our simple models of applying [rewriting] rules to collections of relations, or hypergraphs."

To this author, Wolfram's new work is a cause for celebration. Three hundred years after Leibniz's death, building the world from information and computation continues to be a fertile area for research.

Chapter 3

Complexity & Metamathematics (2023)

Talk delivered on 21 February 2023 at UM6P—Université Mohammed VI Polytechnique in Benguerir, Morocco. Video of talk at https://youtu.be/fBmg_iR2394

What is Metamathematics?

I'd like to talk about metamathematics. Now you may ask what is metamathematics? That's a good question. It's a mysterious subject. So what is this mysterious topic called metamathematics that very few people have heard of?

Well, metamathematics is a part of mathematics but it's also sort of mathematical self-criticism. It's an attempt to psychoanalyze mathematics using mathematical methods, and it's a relatively little-known field.

I'd like to give you an overview of this field as I see it.

This field has a history. I consider Leibniz to be the grandfather of this field, but the father of metamathematics is David Hilbert. And some famous names who worked in this area, famous and not so famous, are Gödel, Turing and Emil Post, who is less well known. Just thought I'd mention their names.

I want to try to give you an overview, some key ideas about what metamathematics is and what it can accomplish.

What is a Formal Axiomatic System (FAS)?

The main object of study in metamathematics is something called a *formal axiomatic system.* It could also be called a *formal axiomatic theory.* And what is that?

Well, you see, if you want to study mathematics using mathematical techniques, how do you do it? You need a sort of a frozen version of mathematics that you can look at from outside, from above, that you can analyze.

And such a frozen version of mathematics, a crystallized version of what mathematics is, is called a formal axiomatic system or a formal axiomatic theory, which would be better but is not the standard name. And what is it? Well, it has a number of components.

You have a finite set of axioms. You have rules of inference, which are the logic that you're using.

The axioms are your postulates, the principles that you start with, and then the idea is to make deductions from the axioms using the rules of inference. And what's unique about a formal axiomatic theory is that the whole system has to be set up so precisely that there are no subjective elements.

In a FAS you're doing mathematical proofs in an artificial language. It's not a programming language. It's a language for reasoning and proving mathematical results, mathematical theorems. And a FAS has to be very precise, so you're not going to use a human language. You're going to be using symbolic logic. The grammar and the syntax are specified very carefully.

In fact the whole thing has to be set up so precisely that there will be a mechanical procedure to check if a proof is correct or not, an algorithm to decide whether or not a proof satisfies the rules. There can be no subjective element. Then you can stand outside the FAS and look at it and ask how well it does, and what it can or cannot accomplish.

That's the metamathematical part, that you're outside the FAS and looking at it and studying it.

The notion of a FAS can be traced back to Leibniz and Llull and probably farther, but it was Hilbert who enunciated the idea of a FAS and the goals of metamathematics most forcefully. He assumed, and it's usually thought to be the case, that if there's one area of human knowledge that gives absolute certainty that area is mathematics. Absolute truth! The presumption is that mathematics is completely black or white or can be if you do it using a formal axiomatic system. Then there's no subjective element, and mathematical reasoning becomes completely objective.

Why did Hilbert propose finding a FAS for all of mathematics? Well, there had been paradoxes and controversies about mathematical methods before Hilbert, and Hilbert's goal was to settle this once and for all by proposing one formal axiomatic system for all of mathematics that mathematicians could agree on, and then in principle there would no longer be any doubt about what are correct mathematical methods.

Hilbert's FAS was to capture all of mathematics, all of mathematical truth, and if you could have a formal axiomatic system for all of mathematics, that would in fact show that mathematics gives absolute truth because a FAS is so precise that there's no uncertainty about whether a proof is correct or not.

So this was the goal, this was the point of departure for metamathematics, but there were a few surprises along the way.

Examples of FAS's

Before talking about these surprises, let me give some examples of formal axiomatic systems or theories.

A nice one that is very well known is Euclidean geometry, but done much more carefully than Euclid did it. Hilbert himself and then Tarski finally completely formalized geometry and made a formal axiomatic theory for geometry. It's not well known that Tarski did some very good work in this area which somehow slipped through the cracks.

Another better known example of a formal axiomatic theory is something called Peano arithmetic. It's just 0, 1, 2, 3, 4, 5, the natural numbers, the positive whole numbers, well, the non-negative whole numbers. And you just have addition, multiplication and equality. It's a very simple system and it includes what mathematicians refer to as number theory, or arithmetic which is the older name. For example, in Peano arithmetic you can prove that there are infinitely many prime numbers, which was demonstrated by Euclid in his Alexandrian compendium of classical Greek mathematics.

Peano arithmetic was studied a great deal in metamathematics, but as a matter of fact there is a more powerful FAS called Zermelo Fraenkel set theory. Set theory starts with Cantor but is finally formalized by the work first of Zermelo and then of Fraenkel. So this is called ZF set theory, and it's known as ZF to its friends.

ZF is a theory which contains a lot more than the non-negative integers. It's a much more powerful system than Peano arithmetic. You can develop, I don't know, most undergraduate math, certainly a lot of mathematics within ZF.

And there are newer examples of formal axiomatic systems as powerful as ZF or even more powerful. So this is the kind of beast that metamathematics likes to study.

Post's Abstract Definition of a FAS

Now in my opinion this notion of a formal axiomatic system or a formal axiomatic theory is too down to earth, too concrete, with a profusion of details that confuse the issue. It was Emil Post who said this most clearly, and he bequeathed us a more abstract definition of a formal axiomatic theory.

In Post's work and in my work the details of what goes on inside the formal axiomatic theory are not that important. I don't really care what kind of logic is used in the theory, what are the rules of inference, nor what are the axioms.

One famous practitioner of metamathematics was Kurt Gödel, and in his work he really needs to dive deep into the bowels of Peano arithmetic and take advantage of what's there. Whereas in Emil Post's approach and in my own, we are flying in an airplane in the stratosphere very high above the formal axiomatic system, so high that the details sort of disappear at that altitude.

So now let me tell you just what Emil Post following Turing, actually, and I following Post perceive to be the essence of a formal axiomatic theory.

Well, Post and I think of a formal axiomatic theory as a program that you put into a computer and out come all the theorems, an infinite set consisting of all the theorems that can be proven within that formal axiomatic theory by making all possible deductions using the rules of inference, using mathematical logic, starting from the postulates or axioms that are your initial principles. In other words, the idea is that you run through all possible proofs and you check which ones are correct, which can be done mechanically because axiomatic theories when they are formalized are very, very precise and there's always a proof-checking algorithm. So you run through all possible proofs in the formal axiomatic system, you check which ones are correct, and that way in principle but not in practice you can generate or calculate or enumerate all the theorems, everything that you can prove in this theory, completely mechanically. There is an algorithm for doing it.

This is the abstract view of what is a formal axiomatic system. It's just an infinite, unending calculation that is going to produce the set of all the truths, all the theorems, that you can prove within the system, all the statements that are provable within that system. This is the object that metamathematics studies. And you can forget what the details are, you can just look at the whole thing from the outside and treat it as a black box.

What is the Complexity of a FAS?

Now I'd like to introduce a notion of complexity that I believe can help us to study, to analyze these formal axiomatic systems. My work has been devoted to applying complexity in metamathematics. It's a very simple-minded notion of complexity. The idea is to measure how big the formal axiomatic system is, how powerful it is, to put it on a scale and weigh it, and here's the way I do that.

I don't need to know what's happening inside the formal axiomatic system. Remember that a formal axiomatic system according to Emil Post is just an algorithm for generating all the theorems, for calculating one by one all the theorems. This algorithm can be programmed out in some programming language, and then the crucial question is to look at the size of this computer program, at how big is the software implementation of the formal axiomatic theory, the software implementation that runs through all possible proofs.

This is a computation that goes on forever, it's very slow, but in principle it can be done if you have a perfectly formalized axiomatic system.

And what interests me is how much software you need, the size of the software implementation of the formal axiomatic system, and I like to measure that in bits rather than bytes, or kilobytes or megabytes.

Hence,

Definition. The *complexity(FAS)* of a formal axiomatic system FAS is defined to be the size in bits of the program *P* that generates, that enumerates, all the theorems of the formal axiomatic system.

P does this by running through all possible proofs and checking which ones are correct. P runs through the tree of all possible deductions from the postulates using all possible applications of the rules of inference. Let me emphasize that this is an infinite computation, this process never ends. One goes on forever looking for theorems that require bigger and bigger proofs, and it's going to be very slow and it's never going to stop, but I don't care, I'm a theoretician. I'm trying to prove theorems, I'm not trying to write software that one can actually run. I'm a pure mathematician, and this is the game I play.

So the key thing here is going to be to measure the power, the complexity of a formal system by the size in bits of the program that calculates all the theorems one by one very, very slowly, but that doesn't matter because I'm only interested in how many bits of information there are. I don't care about the time, which by the way corresponds to the size of the proofs.

This is the key tool that I'm going to use, and it's a notion that Post didn't use, now we are adding something new.

(**Note for Experts:** The program-size complexity measure we are using here is

the old one according to which most N-bit strings have complexity close to N, not the newer self-delimiting program-size complexity measure according to which most N-bit strings have complexity close to $N+$ the self-delimiting program-size complexity of N.)

What is an Elegant Program?

And now I'd like to introduce another concept. Euclidean geometry looks at diagrams of circles and triangles, right?, and Peano arithmetic talks about addition and multiplication, prime numbers, all that kind of thing.

However, the question that I want to consider here is the problem of proving that programs are elegant using formal axiomatic systems. Now, you may ask, what is an elegant program? Well, it's a program, a computer program, and I'm going to call it *elegant* if it has the property that it's the smallest program to calculate what it calculates. More precisely, I like to call a program elegant if it has the property that there is no program written in the same programming language that's smaller and that produces precisely the same output. I don't care if it's faster, or if it's slower, all I care is that it produces the same output.

Definition. An *elegant program* is one with the property that no smaller program written in the same language produces the same output (and also halts or not as the case may be).

It's the smallest program for getting the results that you want, but there might be several elegant programs for the same task, which is why you have to define elegance a little bit more carefully.

I should point out that this notion of elegance is not relevant in software engineering when you are writing real software, because the most elegant program would be rather cryptic. I used to work as a computer programmer for a living and the last thing people who pay programmers want is a program that is incomprehensible. So in the real world, unless you are running a programming competition, elegant programs are not of much interest. But sometimes as a contest people do try to write the most elegant program for something and that can be fun.

Incompleteness: Only Finitely Many Programs are Provably Elegant

The question that I'm now going to consider is using mathematics, or more precisely, using formal axiomatic systems, formal axiomatic theories, to try to prove that programs are elegant. But there are a few facts about elegant programs that you should know before we jump in.

One key fact is that there are infinitely many elegant programs because there are infinitely many different outputs that you may want to write a program to calculate. And as I already said, for every possible output of a computer program, every possible finite or infinite output, there is at least one elegant program, and there may even be several if there's a tie. Another key fact is that there are elegant programs that are arbitrarily big, because there are infinitely many possible outputs that you may want to get from a program, so you're going to need infinitely many programs to do that.

So there are lots of elegant programs out there, tons of them, and what I'd like to consider is the problem of proving that a program is elegant. You have in your hands a specific program and can a formal axiomatic system enable you to prove that it is elegant, that an individual program is an elegant program? It turns out that this is very, very hard to do. It's extremely hard to do. In fact it's so hard to do that it gives you an incompleteness result.

There's a famous incompleteness theorem due to Gödel in 1931, and in 1936 Turing found a different path to incompleteness, and so did Emil Post, and here is yet another path to incompleteness. Formal axiomatic systems are very limited in their ability to prove that individual programs are elegant, in their ability to exhibit specific examples of provably elegant programs. What we shall show is that *you can't prove that a program P is elegant if the size of P in bits is substantially larger than the complexity of the formal axiomatic system you're using to attempt to prove that P is elegant.*

Remember, the complexity of the formal axiomatic system is also measured in bits. It's the size in bits of the program that runs through all possible proofs and generates all possible theorems. And by the way you want to use the same programming language for both the program that you're trying to prove is elegant and for measuring the complexity of the formal axiomatic system.

It's important that you pick one programming language and stick with it throughout this entire discussion. And if I were giving a course on this instead of a lecture I would give more information about the programming languages that are good, that enable you to prove this incompleteness result.

So this will be our key result, and let me point out that, at least to Hilbert, it's disconcerting, pessimistic, and rather unpleasant. As we will soon see, it emerges rather naturally if you consider the complexity of a formal axiomatic system and if you ask if you can prove that a program is elegant. So why is this so disconcerting? Well, to Hilbert it would have been very disconcerting because he wanted to put mathematics on a firm footing by finding one formal axiomatic system for all of mathematics that everyone could agree on, and then you would have very tangible, concrete evidence that mathematics—if it could be done in principle that way, not in practice—provides absolute certainty. This was Hilbert's goal, Hilbert's dream.

But if you have incompleteness, if no formal axiomatic system can capture all of mathematical truth, then you have to keep changing your formal axiomatic system, and where do you get new formal axiomatic systems? Certainty goes out the window because people can argue about whether the postulates of a new formal axiomatic system are correct or self-evident or intuitive. They can also argue about the rules of inference. This pulls the rug out from under Hilbert's proposal to get mathematicians to agree on the rules of the game for pure mathematics, which is very unfortunate.

So using a term that Hilbert never heard of but which physicists now like, there is no theory of everything, no TOE for pure math. That's what incompleteness results show, starting with the famous 1931 incompleteness theorem of Kurt Gödel. Not even if we restrict ourselves to what looks like a not particularly sophisticated mathematical question which is whether or not particular programs are elegant.

Now you may say, what do I care about whether a program is elegant or not? Why bother? The reason I think this approach is interesting is because it's very different from the traditional approach of Kurt Gödel in 1931. The traditional approach was like this: Gödel went from "This statement is false!," which is a paradox, to "This

statement is unprovable!," and then he managed to construct a statement about the whole numbers, a statement in Peano arithmetic, that says it itself is unprovable. Hence this arithmetic statement is true if and only if it's unprovable, and that is an example of incompleteness, it's an assertion about the whole numbers that escapes the power of Peano arithmetic.

Gödel's proof depends on self-reference, on "What I am saying now is false or unprovable," and that looks a bit bizarre. Gödel manages to pull it off, but his true but unprovable arithmetic statement is a little strange. Here we have a different path to incompleteness, involving complexity not self-reference, that I submit is in some ways more fundamental.

A personal anecdote: In 1974 I was living in Buenos Aires and I visited New York and took a chance. I phoned Gödel and he answered the phone himself, believe it or not. I said that I wanted to visit him because I was fascinated by his incompleteness theorem and had a different approach that I wanted to discuss with him. He said, "Well, send me a paper of yours and I'll see if I give you an appointment," and he gave me an appointment. So there I was about to start the trip from Yorktown Heights in New York to Princeton, New Jersey, to the Institute for Advanced Study, and it snowed. There was a snowstorm and Gödel's secretary called me and said that Gödel was very careful about his health and since it had snowed—it wasn't an immense amount of snow—our appointment was canceled. Gödel wasn't coming into the IAS that day because of the snow. So I never got to visit him. The ideas I'm sketching here are basically what I wanted to tell Gödel in 1974. Obviously, in the meantime I've been polishing and improving it a little bit, but the basic message is the same.

So that's why I think these ideas are interesting, because they provide a different and perhaps more fundamental approach to incompleteness.

Now how do you prove this incompleteness result? Well, it's fairly easy. As I said, there is no self-reference. Here is an outline of the proof:

Incompleteness Theorem. *One cannot prove within a given FAS that a program P is elegant if P's size in bits is substantially greater than complexity(FAS).*

Reductio ad absurdum proof. Consider

- the first provably elegant program P (within FAS) that is $2N$ or more bits larger than complexity(FAS).

Here "first" means the first such P that one encounters while enumerating all the theorems of FAS by running through all the possible proofs in FAS and checking which ones are correct.

We wish to show that this P cannot exist for N sufficiently large. Assuming the opposite, note that there is a program only

$$N + complexity(FAS) + c$$

bits in size, namely

- (a self-delimiting fixed main program $c-1$ bits in size) followed by
- (N 1's followed by a 0) followed by
- (the complexity(FAS)-bit program to generate all the theorems of FAS),

that yields the same output as P, by first finding P and then running P.

But this is impossible for N sufficiently large, because

$$N + \mathit{complexity}(FAS) + c \quad < \quad 2N + \mathit{complexity}(FAS)$$

for all N greater than c, contradicting the assumption that P is provably elegant and therefore elegant, assuming that FAS proves only true theorems, which is our tacit hypothesis. ***Q.E.D.***

In a nutshell, the idea of the proof is that if the first provably elegant program P more than

$$2N + \mathit{complexity}(FAS)$$

bits in size were to exist, then there would be a program only

$$N + \mathit{complexity}(FAS) + c$$

bits in size for calculating the exact same output as P produces, which is impossible for N greater than c, because then P would turn out not be an elegant program and FAS proves false theorems, which we are assuming is not the case, because otherwise we would have no interest in studying this FAS.

The basic idea of the proof is simple, but as is usually the case in mathematics, the Devil is in the details.

Completeness: Exhibiting as Many Elegant Programs as Possible

Now I'd like to look at the converse, at the other side of the coin. We used complexity to get an incompleteness result, so there's this low upper bound on the size of provably elegant programs. Now let's look at the converse, at how well you can actually do.

The idea is that if we increase the complexity of the formal axiomatic system maybe we can prove that more programs are elegant and get right up to the upper bound in the incompleteness theorem. And the answer is that you can, and that by increasing the complexity of the formal axiomatic system in the right way, we can get right up to the limit in the incompleteness theorem.

So here is a completeness theorem, that's the other side of the coin. There is a formal axiomatic system which basically just has N bits of complexity that enables you to exhibit every elegant program that halts that is less than N bits in size, which is to say get right up to the upper bound imposed by our incompleteness result.

This is a concrete example of the claim that by increasing the complexity of a formal axiomatic system you can accomplish more. You can also increase the complexity of the formal axiomatic system and not accomplish anything if you add axioms that are not powerful enough to buy you much, but let me show you how it works if you do things in the right way.

There is only one thing that you need to know to determine all the elegant programs that halt that are less than N bits in size. That is the N-bit base-two numeral telling you how many programs halt that are less than N bits in size. This is the axiom, this is the piece of information that you need to know.

Here is the proof:

Completeness Theorem. *There is an* $(N + c')$*-bit FAS,*

$$complexity(FAS) = N + c',$$

that enables one to exhibit every elegant program less than N *bits in size that halts.*

Proof. Consider the N-bit binary numeral giving the number of programs less than N bits in size that eventually halt. If we are given this as our axiom, we first look at the size of this bit string to determine N, then we run all of the less than 2^N programs of size less than N in parallel until exactly that many of these programs halt. In the process, we see what is the output of the programs that halt, and which are the elegant programs that halt that are less than N bits in size. It takes c' bits to program this process, yielding an $N + c'$ bit Post abstract FAS. ***Q.E.D.***

Philosophical Implications: Is Mathematics Quasi-Empirical?

I'm very sorry, I have to apologize for all this technical mathematics, but I'm a mathematician, I like to prove things. So we have two key theorems: an incompleteness theorem and a completeness theorem, both having to do with using formal axiomatic systems to prove that individual programs are elegant.

What does all this imply? I think that incompleteness has philosophical implications.

Gödel's original incompleteness proof "I'm unprovable" was very shocking at first, but then people said "This doesn't have much to do with the kind of math I'm doing." The kind of people I'm talking about are mathematicians, working mathematicians. "Gödel's theorem doesn't have much to do with the theorems I'm interested in. I don't want to prove that I'm unprovable!"

So it's an important question: How seriously should one take incompleteness? Does this really destroy Hilbert's program? Does it really pull the rug out from under mathematical certainty? Does it mean that we should be doing mathematics differently?

In fact, the formal axiomatic system is still a religion in pure math. It's kind of a religious dogma. It's also a practical tool, because there are now some very good formalizations of the mathematics that contemporary mathematicians do.

Some very powerful contemporary mathematicians recently demanded such systems because they were unsure if some of their complicated proofs were correct. They wanted to put their proofs into a formal axiomatic system and check them, they felt they needed reassurance.

Thus there is very good work that continues to build on the idea of a formal axiomatic system in spite of Gödel's incompleteness theorem.

But let me cut to the quick and share with you my contrarian view. Imre Lakatos has a paper where he coins a beautiful term, *quasi-empirical mathematics.* He says that mathematics is quasi-empirical. What exactly does this mean? Well, it means it's not theoretical physics by any means, it's not an experimental science, but perhaps it's not as radically different from an experimental science as people might think.

Vladimir Arnold, who was a fine Russian mathematician, whom I bumped into at a copying machine at the University of Paris one day, stated this in a very colorful way. He

said that "Math is just like physics, except that the experiments are cheaper!," because you use a computer instead of having to have a physics lab full of large, complicated, temperamental equipment. That's how he put it.

Let me talk a little more about this idea, which I think is definitely a minority position in the world of mathematics.

Another way to express the idea that mathematics is quasi-empirical is to say that you believe in *experimental mathematics.* There are a number of books on this by the late Jon Borwein, with additional co-authors and with titles like *Experimentation in Mathematics, Mathematics by Experiment,* and *The Computer as Crucible.* Borwein and his collaborators are real mathematicians, not metamathematicians, they are people who do mathematics for a living, and their notion of experimental mathematics is not as extreme as mine, but nevertheless they sometimes reference my ideas. They believe that after using the computer and doing lots of calculations, you may conjecture a result, but you've still got to prove it using the normally accepted principles. They are not as extreme as I am, because I advocate sometimes basing proofs on new axioms.

I also want to invoke someone else as support for my position that math is quasi-empirical, someone from an earlier generation than Borwein and his collaborators. When I was a student, a very influential mathematician, at least for me, was the Hungarian refugee Pólya who had settled at Stanford University. He wrote a number of books and articles in favor of experimental mathematics, which he referred to as *heuristic or plausible reasoning.*

In addition to being a fine mathematician, Pólya knew a lot about the history of science and mathematics, and in particular about the famous pioneer Leonhard Euler. Pólya wrote a two-volume set *Mathematics and Plausible Reasoning* published by Princeton University Press. Those books contain many case studies of how Euler made his remarkable discoveries. Euler would perform extensive numerical calculations and conjecture a result, and then in a series of papers he would gradually complete a proof, but in some cases never completed the proof. So, according to Pólya, Euler worked a lot like a physicist.

There's also a short article by Pólya which is really beautiful that I loved when I was a young student. It was published in the *American Mathematical Monthly* in 1959 and it's called "Heuristic Reasoning in the Theory of Numbers." He gives an example of a conjecture in the theory of numbers that seems to be the case, and he proves it the way that a physicist might, with a plausible probabilistic argument. I highly recommend this little article.

But I have more radical views than Pólya does. It's true that Gauss and Euler enjoyed calculating, making conjectures and trying to prove them. But I'd like to take a more radical position because I believe, as is suggested by the two theorems I proved here, that sometimes you need to increase the complexity of a formal axiomatic system. And you need to do that by adding new principles, new axioms, that are not self-evident but that are justified by their fertile consequences. You see, there is no way to prove that a new axiom is correct, because if you could do that it would be a theorem, not an axiom.

In my opinion, the way you justify a new mathematical axiom is the way physicists justify the Schrödinger equation. You can't deduce Maxwell's equations or the Schrödinger equation from Newtonian physics. They are justified pragmatically, because of their fertile consequences, because they explain a vast range of new phenomena

that aren't treated in Newtonian physics.

So this is my proposal.

To put it more romantically, I believe in creativity in mathematics, I believe that math isn't a closed system, it's an open system. A fixed formal axiomatic system of the kind that Hilbert wanted for all of math, yes, it would have given us absolute certainty, but it would also have made mathematics into a frozen cemetery.

I believe in adding new axioms, and I've been saying this for many years, and of course nobody is convinced by my arguments, but in fact since I'm not so young anymore, I've actually seen some new axioms prevail. One outstanding example is in abstract set theory, a very difficult subject, and I think it's called the *axiom of projective determinacy (PD).* The set-theoretical community, some very smart mathematicians, have convinced themselves to add PD to ZF because it enables them to prove a lot of results that were conjectured but couldn't be proven, but PD still doesn't enable them to settle Cantor's continuum hypothesis.

Another very hard branch of pure mathematics is number theory or arithmetic, and I used to think that the Riemann hypothesis should be added as a new axiom, because I had seen a number of papers that were "modulo the Riemann hypothesis," which is a mathematical way of saying that they were basing the proofs in those papers on an unproved assumption that is called the Riemann hypothesis. Unfortunately, one day a number theorist explained to me that there are in fact many versions of the Riemann hypothesis, and some are good for this and some are good for that, and he did not feel that there is such a clear compelling case for adding any one of them as a new axiom.

Anyway, number theory is a field with many famous conjectures that empirically, by doing extensive calculations, seem to be the case, but that nobody can prove. For example, there is a conjectured distribution law for the twin primes, a lovely formula that gives a better and better approximation of the number of twin primes as you go farther and farther out. Pólya discusses this in his 1959 paper. Nobody can prove it, but there is a probabilistic heuristic argument, and a physicist would say, well, what more do you want? There's a nice graph, the asymptotic formula fits the graph beautifully, better and better as you go further out, what more could you possibly want? Unfortunately, this is not a particularly good new axiom because it doesn't enable you to prove lots of other results.

Okay, it's time for me to wrap this up. To end with a summary that is also a provocation, I believe that complexity reveals the depth of the incompleteness phenomenon: For the Platonic world of mathematical ideas has infinite complexity, but any FAS only has finite complexity. Therefore, any FAS can only capture an infinitesimal part of the richness of pure mathematics.

Further Reading on Complexity & Metamathematics

- Stephen Wolfram, *Metamathematics: Foundations and Physicalization,* Wolfram Media, 2022.

- Fouad Laroui, *Dieu, les mathématiques, la folie,* Robert Laffont, 2018.

- Réda Benkirane, *Islam, à la reconquête du sens,* Le Pommier, 2017.

- "Complexité, logique et hasard. Entretien avec Gregory Chaitin" in Réda Benkirane, *La Complexité, vertiges et promesses. Histoires de sciences,* Le Pommier, 2002, UM6P-Press 2023. Chapter 1 in this book.

- Essays by Lakatos and Chaitin in Thomas Tymoczko, *New Directions in the Philosophy of Mathematics,* Princeton University Press, 1998. (Includes the paper that Chaitin sent to Gödel in 1974).

- Gregory Chaitin, "Randomness in Arithmetic and the Decline & Fall of Reductionism in Pure Mathematics" in John Cornwell, *Nature's Imagination,* Oxford University Press, 1995.

- Gregory Chaitin, "The Perfect Language," 2015, http://inference-review.com/article/the-perfect-language, Chapter 5 in this book.

Chapter 4

Summer Adventure Seminar Series: Paradoxes of Randomness (2001)

Abstract: I'll discuss how Gödel's paradox "This statement is false/unprovable" yields his famous result on the limits of axiomatic reasoning. I'll contrast that with my work, which is based on the paradox of "The first uninteresting positive whole number", which is itself a rather interesting number, since it is precisely the first uninteresting number. This leads to my first result on the limits of axiomatic reasoning, namely that most numbers are uninteresting or random, but we can never be sure, we can never prove it, in individual cases. And these ideas culminate in my discovery that some mathematical facts are true for no reason, they are true by accident, or at random. In other words, God not only plays dice in physics, but even in pure mathematics, in logic, in the world of pure reason. Sometimes mathematical truth is completely random and has no structure or pattern that we will ever be able to understand. It is **not** the case that simple clear questions have simple clear answers, not even in the world of pure ideas, and much less so in the messy real world of everyday life.

This talk was given 8 August 2001 in the auditorium of the Yorktown lab of the IBM Thomas J. Watson Research Center, which is a part of IBM's Research Division. It was simultaneously viewed on TV in the Hawthorne lab of the Watson Research Center. It was also recorded as digital video. This is an edited transcript of the video. There are no section titles; the displayed material is what I wrote on the transparencies as I spoke.

Hi everybody! I'm delighted to have a chance to talk! Especially in a series with such a delightful name: "summer adventure"! That sounds great!

So some of you are summer visitors of ours? Summer students? **You** look like a summer student! So thanks for coming to visit us. We hope you'll be back. Unfortunately the summer's drawing to a close and I understand many of you are leaving this weekend. What can you do, that's life!

I think it's sort of fun that yesterday Dick Garwin, formerly of this lab, gave a talk, it wasn't exactly a talk on how he built the H-bomb, but it was related to his work in that area. And I thought it was amusing that in our Hawthorne lab building his talk,

which is very important, the poster for it was next to the poster for my talk, which is **un**important, it's on abstract nonsense! And I think that's a large part of what goes on here at Research. So before I launch into my own story, I'd like to tell you about that.

You see, I don't know how much Dick talked about what he's done over the rest of his life. The H-bomb and getting involved in defense, advising about defense, bad defense, good defense, that's clearly very important, and there's a lot of money involved, and it has to do with the real word, the really real world! But what you may not know is that Dick Garwin, Richard Garwin, had a day job and he had a night job! You know, his day job was doing important things like that, but he was also interested in **theory!** One of the things he did at the old Watson lab back in New York City, that then became part of this Research Division, was an experiment designed to check for gravity waves.

A gentleman, I think his name was Weber, had announced in an article on the cover of *Scientific American* that he had detected gravity waves. The story that the man who was the editor in chief of *Scientific American* at that time, Dennis Flanagan, told me, was that he asked Garwin, and Garwin said immediately, "It's impossible! That experiment could not have detected those gravity waves; they're too weak. But just to be sure, I'll repeat the experiment!" Which he did at the IBM Watson lab that used to exist in New York City across the street from Columbia University. He did the experiment very, very carefully, and there was nothing there.

Gravity waves are a very sexy, fundamental intellectual, cosmological fundamental physics question, with no practical applications. And as you [looking at the summer student] reminded me, building an H-bomb is actually nuclear physics, and before the Second World War turned it into a matter of life and death, nuclear physics was about as impractical as writing poetry in ancient Greek! You know, there are people in universities who do that! And there were a few people who were interested in what goes on in the nucleus. And everybody thought that this is crazy and why should one waste one's time like that!

So Richard Garwin was talking about very important practical stuff yesterday, and I would like to talk about stuff which I'm proud to say is very **im**practical! In fact, it's basically philosophy, epistemology, limits of knowledge. And in fact one of my basic theorems that I have in this field, and maybe you could even say that it's the single most important result in the field, is that the field is useless for practical applications! I'll explain this later.

And to give you an idea of the personality of Yorktown Research, let me explain that just like Richard Garwin had a day job and a night job, well I also had what I did for a living and what I did as a hobby. And what I did for a living here at Research was to be working with some gentlemen who are here in the audience now on a wonderful project which was an immense amount of fun, which was we made IBM's UNIX workstation, which was originally called the RS/6000, we designed the prototype for that here at the lab. And then that became, I think, the basis for the processor in Apple's Power Mac.

So just like Richard Garwin, I've also played on both sides of the tennis court: really practical, fun stuff, the real world, making something happen, doing something; and also in the world of ideas. And these worlds do intermix, right? Now, your [the summer student] remark was, maybe my stuff will lead to a quark bomb or something. What kind of a bomb did you say? Right, an anti-matter bomb maybe fifty years from now!

Well, I hope not! But it may well be the case that this kind of stuff could lead to practical applications. In fact, there are people who have actually thought about this already, it's just that it doesn't particularly interest me.

After this general introduction, talking about the kinds of things that go on at Research, let me now launch into my own stuff... So we hope you've had a nice summer here. I remember when I was in high school I visited IBM a few times, and for me it was very exciting, it was like science fiction. So I hope it's been an interesting experience for you and I hope you'll come back!

Okay, so let me launch now into my particular story! The story I'd like to tell you about is as follows. When I was a kid, when I was in high school I was fascinated by computers, which at that time was a little more unusual. I was actually fascinated with programming computers and things like that, on the one hand. On the other hand, I was fascinated with mathematics and physics. And I was fascinated, in particular, with some crazy work of a mathematician called Kurt Gödel, which talks about the limits of mathematical reasoning. That was my theoretical interest, my hobby; my other hobby was computer programming—lucky for me, so I was able to get a job at IBM!

So Kurt Gödel was a hero of mine when I was a child. You know, I thought, how could mathematics prove that mathematics has limitations? How could you use reasoning to show that reasoning has limitations?

And I was lucky, I had a good idea! And I'll explain that idea in a moment. It's a pretty simple idea, and you could have had that idea! And the result of that idea, to skip to the end of the story... Well, I'm not dead yet! [Laughter] Ten years ago... I have a book coming out called *Conversations with a Mathematician*. It's in press now, at this very moment Springer-Verlag is printing it in England. And in this book, *Conversations with a Mathematician*, one of the chapters tells the story of how ten years ago, it was in 1991, I was welcomed in Vienna as Kurt Gödel's successor.

I was invited to give some talks in Vienna. One of the talks was in the classroom where Gödel used to teach, there's a plaque on the wall saying that. You can imagine how thrilled I was! And one of the things that happened when I arrived in Vienna is, they showed me a newspaper published in Vienna, Austria, and there was a full-page article with my photograph in it, and the title of the article was, they translated it to me, "Out-Gödeling Gödel"! So I thought that was a helluva thing, I never expected to have this kind of an experience. Considering that Gödel was for me a god when I was a kid in high school, I thought it was really neat to be welcomed in Vienna this way.

So I'd like to tell you how I did it. So how do you out-Gödel Gödel? Well, what you need is a good idea. All it takes is a good idea! First you've got to get a good idea, and the next thing you've got to do, is you've got to stick with it, work on it, don't give up.

So what was the idea I had? So let me tell you the idea... Well, first let me tell you how Gödel did it, to set the stage a little bit, so then I can compare and contrast the way I do it.

So let's go to how Gödel shows that reasoning has limits. And the way he does it is he uses this paradox:

"This statement is false!"

You have a statement which says of itself that it's false. Or it says

"I'm lying!"

"I'm lying" doesn't sound too bad! But "the statement I'm making now is a lie, what I'm saying right now, this very statement, is a lie", that sounds worse, doesn't it? This is an old paradox that actually goes back to the ancient Greeks, it's the paradox of the liar, and it's also called the Epimenides paradox, that's what you call it if you're a student of ancient Greece.

And looking at it like this, it doesn't seem something serious. I didn't take this seriously. You know, so what! Why should anybody pay any attention to this? Well, Gödel was smart, Gödel showed why this was important. And Gödel changed the paradox, and got a theorem instead of a paradox. So how did he do it? Well, what he did is he made a statement that says of itself,

"This statement is unprovable!"

Now that's a big, big difference, and it totally transforms a game with words, a situation where it's very hard to analyze what's going on. Consider

"This statement is false!"

Is it true, is it false? In either case, whatever you assume, you get into trouble, the opposite has got to be the case. Why? Because if it's true that the statement is false, then it's false. And if it's false that the statement is false, then it's true.

But with

"This statement is unprovable!"

you get a theorem out, you don't get a paradox, you don't get a contradiction. Why? Well, there are two possibilities. With

"This statement is false!"

you can assume it's true, or you can assume it's false. And in each case, it turns out that the opposite is then the case. But with

"This statement is unprovable!"

the two possibilities that you have to consider are different. The two cases are: it's provable, it's unprovable.

So if it's provable, and the statement says it's unprovable, you've got a problem, you're proving something that's false, right? So that would be very embarrassing, and you generally assume by hypothesis that this cannot be the case, because it would really be too awful if mathematics were like that. If mathematics can prove things that are false, then mathematics is in trouble, it's a game that doesn't work, it's totally useless.

So let's assume that mathematics does work. So the other possibility is that this statement

"This statement is unprovable!"

is unprovable, that's the other alternative. Now the statement is unprovable, and the statement says of itself that it's unprovable. Well then it's true, because what it says corresponds to reality. And then there's a hole in mathematics, mathematics is "incomplete", because you've got a true statement that you can't prove. The reason that you have this hole is because the alternative is even worse, the alternative is that you're proving something that's false.

The argument that I've just sketched is not a mathematical proof, let me hasten to say that for those of you who are mathematicians and are beginning to feel horrified that I'm doing everything so loosely. This is just the basic idea. And as you can imagine, it takes some cleverness to make a statement in mathematics that says of itself that it's unprovable. You know, you don't normally have pronouns in mathematics, you have to have an indirect way to make a statement refer to itself. It was a very, very clever piece of work, and this was done by Gödel in 1931.

1931

Okay, so that's the way **he** did it.

Now let me tell you my approach. My approach starts off like this... I'll give you two versions, a simplified version, and a slightly less-of-a-lie version.

The simplified version is, you divide all numbers... you think of whether numbers are interesting or uninteresting, and I'm talking about whole numbers, positive integers,

0, 1, 2, 3, 4, 5, ...

That's the world I'm in, and you talk about whether they're interesting or uninteresting.

Un/Interesting

Somehow you separate them into those that are interesting, and those that are uninteresting, okay? I won't tell you how. Later I'll give you more of a clue, but for now let's just keep it like that.

So, the idea is, then, if somehow you can separate all of the positive integers, the whole numbers, 0, 1, 2, 3, 4, 5, into ones that are interesting and ones that are uninteresting, you know, each number is either interesting or uninteresting, then think about the following whole number, the following positive integer:

"The first uninteresting positive integer"

Now if you think about this number for a while, it's precisely what? You start off with 0, you ask is it interesting or not. If it's interesting, you keep going. Then you look and see if 1 is interesting or not, and precisely when you get to the first uninteresting positive integer, you stop.

But wait a second, isn't that sort of an interesting fact about this positive integer, that it's **precisely** the first uninteresting positive integer?! I mean, it stands out that way, doesn't it? It's sort of an interesting thing about it, the fact that it happens to be precisely the smallest positive integer that's uninteresting! So that begins to give you an idea that there's a problem, that there's a serious problem with this notion of interesting versus uninteresting.

And now you get into a problem with mathematical proof. Because let's assume that somehow you can use mathematics to **prove** whether a number is interesting or

uninteresting. First you've got to give a rigorous definition of this concept, and later I'll explain how that goes. If you can do that, and if you can also prove whether particular positive integers are interesting or uninteresting, you get into trouble. Why? Well, just think about the first positive integer that you can **prove** is uninteresting.

"The first **provably** uninteresting positive integer"

We're in trouble, because the fact that it's precisely the first positive integer that you can **prove** is uninteresting, is a very interesting thing about it! So if there cannot be a first positive integer that you can prove is uninteresting, the conclusion is that you can **never** prove that particular positive integers are uninteresting. Because if you could do that, the first one would *ipso facto* be interesting!

So that's the general idea. But this paradox of whether you can classify whole numbers into uninteresting or interesting ones, that's just a simplified version. Hopefully it's more understandable than what I actually worked with, which is something called the Berry paradox. And what's the Berry paradox?

Berry Paradox

I showed you the paradox of the liar, "This statement is false, I'm lying, what I'm saying right now is a lie, it's false." The Berry paradox talks about

"The first positive integer that can't be named
in less than a billion words"

Or you can make it bytes, characters, whatever, you know, some unit of measure of the size of a piece of text:

Berry Paradox
"The first positive integer that can't be named
in less than a billion words/bytes/characters"

So you use texts in English to name a positive integer. And if you use texts up to a billion words in length, there are only a finite number of them, since there are only a finite number of words in English. Actually we're simplifying, English is constantly changing. But let's assume English is fixed and you don't add words and a dictionary has a finite size. So there are only a finite number of words in English, and therefore if you consider all possible texts with up to a billion words, there are a lot of them, but it's only a finite number, as mathematicians say jokingly in their in-house jargon.

And most texts in English don't name positive integers, you know, they're novels, or they're actually nonsense, gibberish. But if you go through all possible texts of up to a billion words, and there's only a finite list of them, every possible way of using an English text that size to name a number will be there somewhere. And there are only a finite number of numbers that you can name with this finite number of texts, because to name a number means to pick out one specific number, to refer to precisely one of them. But there are an infinite number of positive integers. So most positive integers, almost all of them, require more than a billion words, or any fixed number of words. So just take the first one. Since almost all of them need more than a billion words to be named, just pick the first one.

So this number is there. The only problem is, I just named it in much less than a billion words, even with all the explanation! [Laughter] Thanks for smiling and laughing! If nobody smiles or laughs, it means I blew it, it means that I didn't explain it well! And if everybody laughs, then that's my lucky day!

So there's a problem with this notion of naming, and this is called the Berry paradox. And if you think that the paradox of the liar, "this statement is false", or "what I'm saying now is a lie", is something that you shouldn't take too seriously, well, the Berry paradox was taken even less seriously. I took it seriously though, because the idea I extracted from it is the idea of looking at the size of computer programs, which I call program-size complexity.

Program-Size Complexity

For me the central idea of this paradox is how big a text does it take to name something. And the paradox originally talks about English, but that's much too vague! So to make this into mathematics instead of just being a joke, you have to give a rigorous definition of what language you're using and how something can name something else. So what I do is I pick a computer-programming language instead of using English or any real language, any natural language, I pick a computer-programming language instead. And then what does it mean, how do you name an integer? Well, you name an integer by giving a way to calculate it. A program names an integer if its output is that integer, you know, it outputs that integer, just one, and then it stops. So that's how you name an integer using a program.

And then what about looking at the size of a text measured in billions of words? Well, you don't want to talk about words, that's not a convenient measure of software size. People in fact in practice use megabytes of code, but since I'm a theoretician I use bits. You know, it's just a multiplicative constant conversion factor. [Laughs] In biology the unit is kilobases, right? So every field has its way of measuring information.

Okay, so what does it mean then for a number to be interesting or uninteresting, now that I'm giving you a better idea of what I'm talking about. Well, interesting means it stands out some way from the herd, and uninteresting means it can't be distinguished really, it's sort of an average, typical number, one that isn't worth a second glance. So how do you define that mathematically using this notion of the size of computer programs? Well, it's very simple: a number is uninteresting or algorithmically random or irreducible or incompressible if there's no way to name it that's more concise than just writing out the number directly. That's the idea.

In other words, if the most concise computer program for calculating a number just says to print 123796402, in that case, if that's the best you can do, then that number is uninteresting. And that's typically what happens. On the other hand, if there is a small, concise computer program that calculates the number, that's atypical, that means that it has some quality or characteristic that enables you to pick it out and to compress it into a smaller algorithmic description. So that's unusual, that's an interesting number.

Once you set up this theory properly, it turns out that most numbers, the great majority of positive integers, are uninteresting. You can prove that as a theorem. It's not a hard theorem, it's a counting argument. There can't be a lot of interesting numbers, because there aren't enough concise programs. You know, there are a lot of positive integers, and if you look at programs with the same size in bits, there are only

about as many programs of the same size as there are integers, and if the programs have to be smaller, then there just aren't enough of them to name all of those different positive integers.

So it's very easy to show that **the vast majority** of positive integers cannot be named more concisely than by just exhibiting them directly. Then my key result becomes, that in fact you can never prove it, not in **individual** cases! Even though most positive integers are uninteresting in this precise mathematical sense, you can never be sure, you can never prove it—although there may be a finite number of exceptions. But you can only prove it in a small number of cases. So most positive integers are uninteresting or algorithmically incompressible, but you can almost never be sure in individual cases, even though it's overwhelmingly likely.

That's the kind of "incompleteness result" I get. (That's what you call a result stating that you can't prove something that's true.) And my incompleteness result has a very different flavor than Gödel's incompleteness result, and it leads in a totally different direction.

I gave a talk on this at Bard College a few months ago. And there was a gentleman in the audience—most of them were friendly-looking students—but there was one gentleman in the audience who had a suit and a tie and a white shirt, so I figured he was in a position of authority. It turned out that he was a Dean at Bard College. And at the end of my talk he said, "Well, I have a doctorate in philosophy, and my doctoral thesis was on the paradox of the liar, and I discussed a bunch of paradoxes, but mostly the paradox of the liar, because the Berry paradox is not really considered to be particularly interesting!" I think he meant this remark as a criticism, because as I've explained I base all my work on the Berry paradox. So my reply to this gentleman was, "Yes, that was my great good fortune, that everyone thought that the Berry paradox was uninteresting, that there was this jewel lying in the mud that no one else noticed!" Fortunately no one else thought it was interesting!

Let me mention that I once called up Gödel on the telephone. I was working for IBM in Buenos Aires at that time, and I was visiting Research before I joined Research permanently. This was a long time ago, in 1974. And just before I had to go back to Buenos Aires I called up Gödel on the phone at the Princeton Institute for Advanced Study and I said, "I'm fascinated by your work on incompleteness, and I have a different approach, using the Berry paradox instead of the paradox of the liar, and I'd really like to meet you and tell you about it and get your reaction." And he said, "It doesn't make any difference which paradox you use!" (And his 1931 paper said that too.) I answered, "Yes, but this suggests to me a new information-theoretic view of incompleteness that I'd very much like to tell you about." He said, "Well, send me a paper on this subject and call me back, and I'll see if I give you an appointment."

I had one of my first papers then, actually it was the proofs of one of my first papers on the subject. It was my 1974 *IEEE Transactions on Information Theory* paper; it's reprinted in Tymoczko, *New Directions in the Philosophy of Mathematics*. And I mailed it to him. And I called back. And incredibly enough, he made a small technical remark, and he gave me an appointment. I was delighted, you can imagine, my hero, Kurt Gödel! And the great day arrives, and I'm here in this building, and it was April 1974, Spring. In fact, it was the week before Easter. And I didn't have a car. I was coming from Buenos Aires, I was staying at the YMCA in White Plains and getting a ride here, but I figured out how to get to Princeton, New Jersey by train.

You know, I'd take the train into New York City and then out to Princeton. It would only take me three hours, probably, to do it!

So I'm in my office, ready to go, almost, and the phone rings. And I forgot to tell you, even though it was the week before Easter, it had snowed. These things happen occasionally. It wasn't a whole lot of snow; you know, nothing would stop me from visiting my hero Gödel at Princeton. So anyway, the phone rings, and it's Gödel's secretary, and she says, "Prof. Gödel is extremely careful about his health, and because it's snowed, he's not going to be coming in to the Institute today, so your appointment is canceled!"

And as it happened, that was just two days before I had to take a plane back to Buenos Aires from New York. So I didn't get to meet Gödel! This is another story that I've put in my book *Conversations*...

So all it takes is a new idea! And the new idea was waiting there for anybody to grab it. The other thing you have to do when you have a new idea is, don't give up too soon. As George Polya put it in his book *How to Solve It*, theorems are like mushrooms, usually where there's one, others will crop up! In other words, another way to put it, is that usually the difference between a professional, expert mathematician with lots of experience and a young, neophyte mathematician is not that the older mathematician has more ideas. In fact, the opposite is usually the case. It's usually the kids that have all the fresh ideas! It's that the professional knows how to take more advantage of the few ideas he has. And one of the things you do, is you don't give up on an idea until you get all the milk, all the juice out of it!

So what I'm trying to lead up to is that even though I had an article in *Scientific American* in 1975 about the result I just told you, that most numbers are random, algorithmically random, but you can never prove it, I didn't give up, I kept thinking about it. And sure enough, it turned out that there was another major result there, that I described in my article in *Scientific American* in 1988. Let me try to give you the general idea.

The conclusion is that

Some mathematical facts
are true for no reason,
they're true by accident!

Let me just explain what this means, and then I'll try to give an idea of how I arrived at this surprising conclusion. The normal idea of mathematics is that if something is true it's true for a reason, right? The reason something is true is called a proof. And a simple version of what mathematicians do for a living is they find proofs, they find the reason that something is true.

Okay, what I was able to find, or construct, is a funny area of pure mathematics where things are true for no reason, they're true by accident. And that's why you can never find out what's going on, you can never prove what's going on. More precisely, what I found in pure mathematics is a way to model or imitate, independent tosses of a fair coin. It's a place where God plays dice with mathematical truth. It consists of mathematical facts which are so delicately balanced between being true or false that we're never going to know, and so you might as well toss a coin. You can't do better than tossing a coin. Which means the chance is half you're going to get it right if you toss the coin and half you'll get it wrong, and you can't really do better than that.

So how do I find this complete lack of structure in an area of pure mathematics? Let me try to give you a quick summary. For those of you who may have heard about it, this is what I like to call Omega, it's a real number, the halting probability.

Omega Number
"Halting Probability"

And some people are nice enough to call this "Chaitin's number". I call it Omega. So let me try to give you an idea of how you get to this number.

Well, the way you start getting to this number that shows that some mathematical facts are true for no reason, they're only true by accident, is you start with an idea that goes back to Borel almost a century ago [1927 according to Tasic, *Mathematics and the Roots of Postmodern Thought*], of using one real number to answer all possible yes/no questions, not just mathematical questions, all possible yes/no questions in English—and in Borel's case it was questions in French. How do you do it?

Well, the idea is you write a list of all possible questions. You make a list of all possible questions, in English, or in French. A first, a second, a third, a fourth, a fifth:

Question # 1
Question # 2
Question # 3
Question # 4
Question # 5

The general idea is you order questions say by size, and within questions of the same size, in some arbitrary alphabetical order. You number all possible questions.

And then you define a real number, Borel's number, it's defined like this:

Borel's Number
$.d_1d_2d_3d_4d_5$
The Nth digit after the decimal point, d_N,
answers the Nth question!!

Well, you may say, most of these questions are going to be garbage probably, if you take all possible texts from the English alphabet, or French alphabet. Yes, but a digit has ten possibilities, so you can let 1 mean the answer is true, 2 mean the answer is false, and 3 mean it's not a valid yes/no question in English, because it's not valid English, or it is valid English, but it's not a question, or it is a valid question, but it's not a yes/no question, for example, it asks for your opinion. There are various ways to deal with all of this.

So you can do all this with one real number—and a real number is a number that's measured with infinite precision, with an infinite number of digits d_N after the decimal point—you can give the answers to all yes/no questions! And these will be questions about history, questions about philosophy, questions about what will IBM stock do this Friday, will it go up or down, will it go up by more than five dollars this Friday?! There are a lot of interesting questions!—Probably you have to give the date, because saying "this Friday" is relative.

It'll answer little questions like was Shakespeare really Bacon?! There's an awful lot you can put into a real number. It has an infinite amount of information, because it has

an infinite number of digits. So this is a way to say that real numbers are very **un**real, right? So let's start with this very unreal number that answers all yes/no questions, and I'll get to my Omega number in a few steps.

The next step is to make it only answer questions about Turing's halting problem. So what's Turing's halting problem? Well, the halting problem is a famous question that Turing considered in 1936. It's about as famous as Gödel's 1931 work, but it's different.

Turing's Halting Problem 1936
[1931 Gödel]

And what Turing showed is that there are limits to mathematical reasoning, but he did it very differently from Gödel, he found something concrete. He doesn't say "this statement is unprovable" like Gödel, he found something concrete that mathematical reasoning can't do: it can't settle in advance whether a computer program will ever halt. This is the halting problem, and it's in a wonderful paper, it's the beginning of theoretical computer science, and it was done before there were computers. And this is the Turing who then went on and did important things in cryptography during the Second World War, so Turing was also a jack of all trades, like Richard Garwin.

So how do you prove Turing's result that there's no algorithm to decide if a computer program—a self-contained computer program—will ever halt? (Actually the problem is to decide that it will **never** halt.) Well, that's not hard to do, in many different ways, and I even discovered a Berry paradox, program-size style proof. You might enjoy discovering it yourself. Or you can find it in my last lecture in *Conversations*...

So let's take Borel's real number, and let's change it so that it only answers instances of the halting problem. So you just find a way of numbering all possible computer programs, you pick some fixed language, and you number all programs somehow: first program, second program, third program, you make a list of all possible computer programs in your mind, it's a mental fantasy.

Computer Program # 1
Computer Program # 2
Computer Program # 3
Computer Program # 4
Computer Program # 5

And then what you do is you define a real number whose Nth digit—well, let's make it binary now instead of decimal—whose Nth bit tells us if the Nth computer program ever halts.

Turing's Number
$.b_1b_2b_3b_4b_5$
The Nth bit after the binary point, b_N,
tells us if the Nth computer program ever halts.

So we've already economized a little, we've gone from a decimal number to a binary number. This number is between zero and one, and so is Borel's number, there's no integer part to this real number. It's all in the fractional part. You have an infinite number of digits or bits after the decimal point or the binary point. In the previous

number, Borel's original one, the Nth digit answers the Nth yes/no question in French. And here the Nth bit of this new number, Turing's number, will be 0 if the Nth computer program never halts, and it'll be 1 if the Nth computer program does eventually halt.

So this one number would answer all instances of Turing's halting problem. And this number is uncomputable, Turing showed that in his 1936 paper. There's no way to calculate this number, it's an uncomputable real number, because the halting problem is unsolvable. This is shown by Turing in his paper.

So what's the next step? This still doesn't quite get you to randomness. This number gets you to uncomputability. But it turns out this number, Turing's number, is redundant. Why is it redundant?

Redundant

Well, the answer is that there's a lot of repeated information in the bits of this number. We can actually compress it more, we don't have complete randomness yet. Why is there a lot of redundancy? Why is there a lot of repeated information in the bits of this number? Well, because different cases of the halting problem are connected. These bits b_N are not independent of each other. Why?

Well, let's say you have K instances of the halting problem. That is to say, somebody gives you K computer programs and asks you to determine in each case, does it halt or not.

K instances of the halting problem?

Is this K bits of mathematical information? K instances of the halting problem will give us K bits of Turing's number. Are these K bits independent pieces of information? Well, the answer is no, they never are. Why not? Because you don't really need to know K yes/no answers, it's not really K full bits of information. There's a lot less information. It can be compressed. Why?

Well, the answer is very simple. If you have to ask God or an oracle that answers yes/no questions, you don't really need to ask K questions to the oracle, you don't need to bother God that much! You really only need to know what? Well, it's sufficient to know **how many** of the programs halt.

And this is going to be a number between zero and K, a number that's between zero and K.

$$0 \leq \#\text{ that halt} \leq K$$

And if you write this number in binary it's really only about $\log_2 K$ bits.

$$\#\text{ that halt} = \log_2 K \text{ bits}$$

If you know how many of these K programs halt, then what you do is you just start running them all in parallel until you find that precisely that number of programs have halted, and at that point you can stop, because you know the other ones will never halt. And knowing how many of them halt is a lot less than K bits of information, it's really only about $\log_2 K$ bits, it's the number of bits you need to be able to express a number between zero and K in binary, you see.

So different instances of the halting problem are never independent, there's a lot of redundant information, and Turing's number has a lot of redundancy. But essentially just by using this idea of telling how many of them halt, you can squeeze out all the redundancy. You know, the way to get to randomness is to remove redundancy! You distill it, you concentrate it, you crystallize it. So what you do is essentially you just take advantage of this observation—it's a little more sophisticated than that—and what you get is my halting probability.

So let me write down an expression for it. It's defined like this:

Omega Number

$$\Omega = \sum_{p \text{ halts}} 2^{-|p|}$$

$$|p| = \text{size in bits of program } p$$

$$0 < \Omega < 1$$

Then write Ω in binary!

So this is how you get randomness, this is how you show that there are facts that are true for no reason in pure math. You define this number Ω, and to explain this I would take a long time and I don't have it, so this is just a tease!

For more information you can go to my books. I actually have **four** books published by Springer-Verlag on this subject. The one, *Conversations*, which hasn't quite been published yet, is non-technical. The others get more and more technical. So you can read as many or as few of them as you like. My website has information on all of them.

So you define this number to be what? You pick a computer programming language, and you look at all programs p that halt, p is a program, and you sum over all programs p that halt. And what do you sum? Well, if the program p is K bits long, it contributes $1/2^K$, one over two to the K, to this halting probability.

In other words, each K-bit program has probability $1/2^K$, and you'll immediately notice that there are two to the thousand thousand-bit programs, so probably this sum will diverge and give infinity, if you're not careful. And the answer is yes, you're right if you worry about that. So you have to be careful to do things right, and the basic idea is that no extension of a valid program is a valid program. And if you stipulate that the programming language is like that, that its programs are "self-delimiting", then this sum is in fact between zero and one and everything works. Okay?

Anyway, I don't want to go into the details because I don't have time. So if you do everything right, this sum

$$\sum_{p \text{ halts}} 2^{-|p|}$$

actually converges to a number between zero and one which is the halting probability Ω. This is the probability that a program, each bit of which is generated by an independent toss of a fair coin, eventually halts. And it's a way of summarizing all instances of the halting problem in one real number and doing it so cleverly that there's no redundancy.

So if you take this number and then you write it in binary, this halting probability, it turns out that those bits of this number written in binary, these are independent,

irreducible mathematical facts, there's absolutely no structure. Even though there's a simple mathematical definition of Ω, those bits, if you could see them, could not be distinguished from independent tosses of a fair coin. There is no mathematical structure that you would ever be able to detect with a computer, there's no algorithmic pattern, there's no structure that you can capture with mathematical proofs—even though Ω has a simple mathematical definition. It's incompressible mathematical information. And the reason is, because if you knew the first N bits of this number Ω, it would solve the halting problem for all programs up to N bits in size, it would enable you to answer the halting problem for all programs p up to N bits in size. That's how you prove that this Ω number is random in the sense I explained before of being algorithmically incompressible information.

And that means that not only you can't compress it into a smaller algorithm, you can't compress it into fewer bits of axioms. So if you wanted to be able to determine K bits of Ω, you'd need K bits of axioms to be able to prove what K bits of this number are. It has—its bits have—no structure or pattern that we are capable of seeing.

However, you **can** prove all kinds of nice mathematical theorems about this Ω number. Even though it's a specific real number, it really mimics independent tosses of a fair coin. So for example you can prove that 0's and 1's happen in the limit exactly fifty percent of the time, each of them. You can prove all kinds of **statistical properties**, but you can't determine **individual bits**!

So this is the strongest version I can come up with of an incompleteness result...

In fact, it now looks like I'm going to end up publishing **two** books this year. The new one is called

Conversations with a Mathematician.

That should be out in England in October, and here, not until the end of the year probably, until the copies get here. But I had a previous book, which is called

Exploring Randomness,

that became available in the US at the beginning of the year. And it's a very technical book. These are the least technical and the most technical of my

Springer-Verlag

trilogy, which now has four books in it, if you include *Conversations*! [Laugher]

And the day before yesterday I got fan mail. You know, most of us only get e-mail. So when you get physical mail something funny has happened. So I get an envelope in my office, it's written in pencil, in block letters, and in my experience letters like that are usually written by high-school students. And sure enough this was a letter from someone in the Bronx, that I could have written when I had his age. He said he just got a copy of *Exploring Randomness* and he really should write me this letter after reading it, but he's so excited he wants to write me right away, and the question in the letter is, what would Gödel and Turing think?!

And that's a good question! And I said, "Well, I don't know, what do **you** think they would think?" was the answer I sent him, that's the sage reply. But having had more time to think about it, let me give a better answer here than the one I gave in the letter I sent him.

How do you feel when you spend your life working in an area and say a twenty-year old calls up and does better? And this kind of thing can happen to anybody, you know, Bertrand Russell did it to Frege... How does one feel? Well, it happens in a sense to every parent, right, you have children. I mean, what is better, to be a dead-end or to be a stepping stone?! I think I prefer to be a stepping stone, than to be a dead-end, frankly! So I'd like to suggest some questions that I don't know how to answer, that are connected, that I think are connected with this stuff that I've been talking about. So let me mention some questions I **don't** know how to answer. But I hope that maybe my stuff is a stepping stone.

Well, one question is positive results on mathematics:

Positive Results
Where do new mathematical concepts come from?

I mean, Gödel's work, Turing's work and my work are negative in a way, they're incompleteness results, but on the other hand, they're positive, because in each case you introduce a new concept: incompleteness, uncomputability and algorithmic randomness. So in a sense they're examples that mathematics goes forward by introducing new concepts! So how about an optimistic theory instead of negative results about the limits of mathematical reasoning? In fact, these negative metamathematical results are taking place in a century which is a tremendous, spectacular success for mathematics, mathematics is advancing by leaps and bounds. So there's no reason for pessimism. So what we need is a more realistic theory that gives us a better idea of **why** mathematics is doing so splendidly, which it is. But I'd like to have some theoretical understanding of this, not just anecdotal evidence, like the book about the Wiles proof of Fermat's result. [Singh, *Fermat's Enigma*]

So this is one thing that I don't know how to do and I hope somebody will do.

Another thing which I think is connected, isn't where new mathematical ideas come from, it's where do new biological organisms come from. I want a theory of evolution, biological evolution.

Biological Evolution
Where do new biological ideas come from?

You see, in a way biological organisms are ideas, or genes are ideas. And good ideas get reused. You know, it's programming, in a way, biology.

Another question isn't theoretical evolutionary biology—which doesn't exist, but that is what I'd like to see—another question is where do new ideas come from, not just in math! Our new ideas. How does the brain work? How does the mind work? Where do new ideas come from? So to answer that, you need to solve the problem of AI or how the brain works!

AI/Brain/Mind
Where do new ideas come from?

In a sense, where new mathematical concepts come from is related to this, and so is the question of the origin of new biological ideas, new genes, new ideas for building organisms—and the ideas keep getting reused. That's how biology seems to work. Nature is a cobbler!—So I think these problems are connected, and I hope they have

something to do with the ideas I mentioned, my ideas, maybe, I'm hoping. Perhaps not in the form that I worked on them, but my hope is to be, my fondest hope would be to be a stepping stone.

So I don't know how to answer these questions, but maybe some of you, some of you summer visitors will be able to answer them. That would be my hope. Anyway my final message for you summer visitors—in addition to the hope that you had fun here, and you enjoyed yourselves and maybe you'll come back—is, to sound like an old man, let me just say, well, the future is yours! And do great things!

Thank you! [Applause]

[Twelve minutes of questions and answers]

I'm a bit of a rebel, so when I was the age of you summer visitors, you summer kids, what I did is I went marching off orthogonal to the rest of the human race. I said to myself, "You know, it's no fun to work in somebody else's field, I'll try to make my own field!" So hopefully some of you will do the same and come and talk about it to our summer visitors twenty or thirty years from now!

I think we've run out of time... Thank you! [Applause]

Chapter 5

The Perfect Language (2015)

What follows is based on a talk originally given by the author at the Hebrew University in Jerusalem and then, in expanded form, at the Perimeter Institute in Canada.

I'm going to talk about mathematics, and I'd like to give you a broad overview, most definitely a non-standard view of some intellectual history.

There is a wonderful book by Umberto Eco called *The Search for the Perfect Language*, and I recommend it highly.

In *The Search for the Perfect Language* you can see that Umberto Eco likes the Middle Ages—I think he probably wishes we were still there. The book talks about a dream that Eco believes played a fundamental role in European intellectual history, which is the search for the perfect language.

What is the search for the perfect language? Nowadays a physicist would call this the search for a Theory of Everything (TOE), but in the terms in which it was formulated originally, it was the idea of finding, shall we say, the language of creation, the language before the Tower of Babel, the language that God used in creating the universe, the language whose structure directly expresses the structure of the world, the language in which concepts are expressed in their direct, original format.

You can see that this idea is a little bit like the attempt to find a foundational Theory of Everything in physics.

The crucial point is that knowing this language would be like having a key to universal knowledge. If you're a theologian, it would bring you closer, very close, to God's thoughts, which is dangerous. If you're a magician, it would give you magical powers. If you're a linguist, it would tell you the original, pure, uncorrupted language from which all languages descend.

This very fascinating book is about the quest to find that language. If you find it, you're opening a door to absolute knowledge, to God, to the ultimate nature of reality.

And there are a lot of interesting chapters in this intellectual history, one of them involves a Catalan, Raymond Lull, who lived in or about 1200.

He was a very interesting man who had the idea of mechanically combining all possible concepts to get new knowledge. So you would have a wheel with different concepts on it, and another wheel with other concepts on it, and you would rotate them to get all possible combinations. This would be a systematic way to discover new concepts and new truths. If you remember Jonathan Swift's *Gulliver's Travels*, Swift makes fun of an idea like this.

In *The Search for the Perfect Language*, there is an entire chapter about Gottfried Wilhelm Leibniz. Leibniz is wonderful because he is universal. He knew all about Kabbalah, Christian Kabbalah and Jewish Kabbalah, and all kinds of hermetic and esoteric doctrines, and he knew all about alchemy, he actually ghost-authored a book on alchemy. Leibniz knew about all these things, and he knew about ancient philosophy, he knew about scholastic philosophy, and he also knew about what was then called mechanical philosophy, which was the beginning of modern science. And Leibniz saw good in all of this.

Leibniz formulated a version of the search for the perfect language, which was firmly grounded in the magical, theological original idea, but which is also fit for consumption nowadays, that is, acceptable to modern ears, to contemporary scientists. This is a universal language, which he called the *characteristica universalis*, that was supposed to come with a crucial *calculus ratiocinator*.

The idea is to reduce reasoning to calculation, to computation, because the most certain thing is that $2+5=7$, and what is this if not a calculation? If two people have an intellectual dispute, Leibniz remarked, instead of dueling they could just sit down and say, "Gentlemen, let us compute!" and get the correct answer, and find out who was right.

This is Leibniz's version of the search for the perfect language. How far did he get with it? Well, Leibniz was a person who got bored easily and flew like a butterfly from field to field, throwing out fundamental ideas, rarely taking the trouble to develop them fully.

One case of the *characteristica universalis* that Leibniz *did* develop is called the calculus. This is one case where Leibniz worked out his ideas for the perfect language in beautiful detail.

Leibniz's version of the calculus differs from Isaac Newton's precisely because it was part of Leibniz's project for the *characteristica universalis*. Christiaan Huygens hated the calculus. He taught Leibniz mathematics in Paris at a relatively late age, when Leibniz was in his twenties. Most mathematicians start very, very young. And Christiaan Huygens hated Leibniz's calculus because he said that it was mechanical, it was brainless: any fool can just calculate the answer by following the rules, without understanding what he or she is doing.

Huygens preferred the old, synthetic geometry proofs, where you had to be creative and come up with a diagram and some particular reason for something to be true. Leibniz wanted a general method. He wanted to get the formalism, the notation, right, and have a mechanical way to get the answer. Huygens didn't like this, but that was precisely the point. That was precisely what Leibniz was looking for. The idea was that if you get absolute truth, if you have found the truth, it should mechanically enable you to determine what's going on, without creativity. This is good, this is not bad. This is also precisely how Leibniz's version of the calculus differed from Newton's. Leibniz saw clearly the importance of having a formalism that led you automatically to the answer.

Let's now take a big jump, to David Hilbert, about a century ago. No, first I want to tell you about an important attempt to find the perfect language: Georg Cantor's theory of infinite sets. This late nineteenth-century theory is interesting because it's firmly based in the Middle Ages and also, in a way, the inspiration for all of twentieth-century mathematics. This theory of infinite sets was actually theology—*mathematical* theology. Normally you don't mention that fact. The price of admission to the field

of mathematics demands that the mathematician throw out all the philosophy, leaving only something technical behind. So all the theology has been thrown out.

But Cantor's goal was to understand God. God is transcendent. The theory of infinite sets has a hierarchy of bigger and bigger infinities, the alephs, the ℵ's. You have $\aleph_0$, $\aleph_1$, the infinity of integers, of real numbers, and you keep going. Each one of these is the set of all subsets of the previous one. And very far out you get mind-boggling infinities like $\aleph_\omega$. This is the first infinity after

$$\aleph_0, \aleph_1, \aleph_2, \aleph_3, \aleph_4 \ldots$$

Then you can continue with

$$\omega + 1, \omega + 2, \omega + 3 \ldots 2\omega + 1, 2\omega + 2, 2\omega + 3 \ldots$$

These so-called ordinal numbers are subscripts for the ℵ's, which are cardinalities. Let's go farther:

$$\aleph_{\omega^2}, \aleph_{\omega^\omega}, \aleph_{\omega^{\omega^\omega}} \ldots$$

and there's an ordinal called epsilon-nought

$$\epsilon_0 = \omega^{\omega^{\omega^{\omega^{\cdots}}}}$$

which is the smallest solution of the equation

$$x = \omega^x$$

The corresponding cardinal

$$\aleph_{\epsilon_0}$$

is pretty big!

God is very far off, since God is infinite and transcendent. We can try to go in his direction. But we're never going to get there, because after every cardinal, there's a bigger one, the cardinality of the set of all subsets. And after any infinite sequence of cardinals that you get, you just take the union of all of that, and you get a bigger cardinal than is in the sequence. So this thing is inherently open-ended.

This is absolutely wonderful, breathtaking stuff.

The only problem is that it's contradictory.

The problem is very simple. If you take the universal set, the set of everything, and you consider the set of all its subsets, by Cantor's diagonal argument this should have a bigger cardinality, but how can you have anything bigger than the set of everything?

This is the paradox that Bertrand Russell discovered. Russell looked at this and asked why you get this bad result. And if you look at the Cantor diagonal argument proof that the set of all subsets of everything is bigger than everything, it involves the set of all sets that are not members of themselves,

$$\{x : x \notin x\},$$

which can neither be in itself nor not be in itself. This is called the Russell paradox.

Cantor was aware that this happens, but he wasn't bothered by these contradictions, because he was doing theology. We're finite but God is infinite, and it's paradoxical

for a finite being to try to comprehend a transcendent, infinite being, so paradoxes are fine. But the mathematical community was not very happy with a theory which leads to contradictions. What mathematicians have done is forget about all this theology and philosophy and try to sweep the contradictions under the rug. There is an expurgated version of all this called Zermelo–Fraenkel set theory, with the axiom of choice, usually designated ZFC.

This is a formal axiomatic theory that you develop using first-order logic, and it is an expurgated version of Cantor's theory believed not to contain any paradoxes.

Bertrand Russell was inspired by all of this to attempt a general critique of mathematical reasoning, and to find a lot of contradictions, a lot of mathematical arguments that lead to contradictions. I already told you about his most famous one, the Russell paradox.

Russell was an atheist who was searching for the absolute, who believed in absolute truth. And he loved mathematics and wanted mathematics to be perfect. Russell went around telling people about these contradictions in order to try to get them fixed.

Besides the paradox that there's no biggest cardinal, and that the set of subsets of everything is bigger than everything, there's also a problem with the ordinal numbers that's called the Burali–Forti paradox, namely that the set of all the ordinals is an ordinal that's bigger than all the ordinals. This works because each ordinal can be defined as the set of all the ordinals that are smaller than it is. (Then an ordinal is less than another ordinal if and only if it is contained in it.)

Russell was going around telling people that reason leads to contradictions. So David Hilbert, about a century ago, proposed a program to put mathematics on a firm foundation. And basically what Hilbert proposed is the idea of a completely formal axiomatic theory, which is a modern version of Leibniz's *characteristica universalis* and *calculus ratiocinator*.

In such a formal axiomatic theory you would have a finite number of axioms, axioms that are not written in an ambiguous natural language. Instead you use a precise artificial language with a simple, regular artificial grammar. You use mathematical logic, not informal reasoning, and you specify the rules of the game precisely. It should be mechanical to decide whether a proof is correct.

Hilbert was a conservative. He believed that mathematics gives absolute truth, which is an idea from the Middle Ages. You can see evidence of the Middle Ages whenever you mention absolute truth. Nevertheless, modern mathematicians remain enamored with absolute truth. As Kurt Gödel said, we pure mathematicians are the last holdouts of the Middle Ages. We still believe in the Platonic world of ideas, at least mathematical ideas, when everyone else, including philosophers, now laughs at this notion. But pure mathematicians live in the Platonic world of ideas, even though everyone else stopped believing in it a long time ago.

So mathematics gives absolute truth, said Hilbert. Every mathematician somewhere deep inside believes this. Then there ought to exist a finite set of axioms, and a precise set of rules for deduction, for inference, such that all mathematical truth is a consequence of these axioms. You see, if mathematical truth is black or white, and purely objective, then if you fill in all the steps in a proof and carefully use an artificial language to avoid ambiguity, you should be able to have a finite set of axioms we can all agree on, that in principle enables you to deduce all mathematical truth. This is just the notion that mathematics provides absolute certainty.

An important consequence of this idea goes back to the Middle Ages. This perfect language for mathematics, which is what Hilbert was looking for, would in fact give a key to absolute knowledge, because in principle you could mechanically deduce all the theorems from the axioms, simply by running through the tree of all possible proofs. You start with the axioms, then you apply the rules of inference once, and get all the theorems that have one-step proofs; you apply them two times, and you get all the theorems that have two-step proofs; and like that, totally mechanically, you would get all of mathematical truth, by systematically traversing the tree of all possible proofs.

This would not put all mathematicians out of work. In practice this process would take an outrageous amount of time to get to interesting results, and all the interesting theorems would be overwhelmed by uninteresting theorems, such as the fact that $1+1 = 2$. It would be hard to find the interesting theorems and to separate the wheat from the chaff. But in principle this would give you all mathematical truths. You wouldn't actually do it, but it would show that mathematics gives absolute certainty.

So this was the idea of putting mathematics on a firm foundation and removing all doubts. This was Hilbert's idea, about a century ago. Meta-mathematics studies a formal axiomatic theory from the outside. Notice that this is a door to absolute truth, following the notion of the perfect language.

What happened? There is some good news and some bad news. Some of the good news I already mentioned. The thing that comes the closest to what Hilbert asked for is Zermelo–Fraenkel set theory, and it is a beautiful axiomatic theory. I want to mention some of the milestones in the development of this theory. One of them is the von Neumann integers, so let me tell you about that. Remember that Baruch Spinoza had a philosophical system in which the world is built out of only one substance, and that substance is God, that's all there is. Zermelo–Fraenkel set theory is similar. Everything is sets, and every set is built out of the empty set. That's all there is: the empty set, and sets built starting with the empty set.

Zero is the empty set $\{\}$, that's the first von Neumann integer, and in general $n+1$ is defined to be the set of all integers less than or equal to n:

$$n + 1 = \{0, 1, 2, \ldots, n\}.$$

If you write this out in full, removing all the abbreviations, all you have are curly braces, you have set formation starting with no content, and the full notation for n grows exponentially in n because everything up to that point is repeated in the next number. In spite of this exponential growth, this is a beautiful conceptual scheme.

Then you can define rational numbers as pairs of these integers, you can define real numbers as limit sequences of rational numbers, and you get all of mathematics, starting just with the empty set. So it's a lovely piece of ontology. Here's all of mathematical creation just built out of the empty set.

This is a formal theory that most mathematicians believe enables you to carry out all the arguments that normally appear in mathematics—maybe if you don't include category theory, which is very difficult to formalize, and even more paradoxical than set theory, from what I hear.

So that's some of the positive work on Hilbert's program. Now some of the negative work on Hilbert's program is, of course, Gödel in 1931 and Alan Turing in 1936. What they show is that you can't have a perfect language for mathematics, you cannot have a formal axiomatic theory for all of mathematics because of incompleteness, because

no such system will include all of mathematical truth. It will always leave out truths; it will always be incomplete.

This is Gödel's incompleteness theorem of 1931, and Gödel's original proof is very strange. It's basically the paradox of "this statement is false," which is a paradox, of course, because it can be neither true nor false. If it's false that it's false, then it's true, and if it's true that it's false, then it's false. That's just a paradox. But what Gödel does is say "this statement is unprovable." So if the statement says of itself it's unprovable, there are two possibilities: it's provable, or it isn't. If it's provable, then we're proving something that's false, because it says it's unprovable. So we hope that's not the case; by hypothesis, we'll eliminate that possibility. If we prove things that are false, we have a formal axiomatic theory that we're not interested in, because it proves false things. The only possibility left is that it's unprovable. But if it's unprovable then it's true, because it asserts it's unprovable, therefore there's a hole. We haven't captured all of mathematical truth in our theory.

This proof of incompleteness shocked a lot of people.

A better proof of incompleteness, a deeper proof, comes from Turing in 1936. He derived incompleteness from a more fundamental phenomenon, which is uncomputability, the discovery that mathematics is full of stuff that can't be calculated, of things you can define, but which you cannot calculate, because there's no algorithm. And in particular, the uncomputable thing that he discovered was the halting problem, a very simple question: does a computer program that's self-contained halt, or does it go on forever? There is no algorithm to answer this in every individual case, therefore there is no formal axiomatic theory that enables you to always prove in individual cases what the answer is.

So Turing's insight in 1936 was that incompleteness, that Gödel found in 1931, for any formal axiomatic theory, comes from a deeper phenomenon, which is uncomputability. Incompleteness is an immediate corollary of uncomputability, a concept which does not appear in Gödel's 1931 paper.

But Turing's paper has both good and bad aspects. There's a negative aspect of his 1936 paper, which I've just told you about, but there's also a positive aspect. You get another proof, a deeper proof of incompleteness, but you also get a kind of completeness.

You find a perfect language.

There is no perfect language for mathematical reasoning. Gödel showed that in 1931, and Turing showed it again in 1936. But what Turing also showed in 1936 is that there are perfect languages, not for mathematical reasoning, but for computation, for specifying algorithms. What Turing discovered in 1936 is that there's a kind of completeness called universality and that there are universal Turing machines and universal programming languages.

What universal means, what a universal programming language or a universal Turing machine is, is a language in which every possible algorithm can be written. On the one hand, Turing showed us in a deeper way that any language for mathematical reasoning has to be incomplete, but on the other hand, he showed us that languages for computation can be universal, which is just a synonym for completeness. There are perfect languages for computation, for writing algorithms, even though there aren't any perfect languages for mathematical reasoning.

This is the positive side, this is the completeness side, of Turing's 1936 paper.

Now, what I've spent most of my professional life on, is a subject I call algorithmic

information theory, which derives incompleteness from uncomputability by taking advantage of a deeper phenomenon, by considering an extreme form of uncomputability, which is called algorithmic randomness or algorithmic irreducibility.

There's a perfect language again, and there's also a negative side, the halting probability Ω, whose bits are algorithmically random, algorithmically irreducible mathematical truths.

$$\Omega = .010010111\ldots$$

This is a place in pure mathematics where there's no structure. If you want to know the bits of the numerical value of the halting probability, this is a well-defined mathematical question, and in the world of mathematics all truths are necessary truths, but these look like accidental, contingent truths. They look random; they have irreducible complexity.

There are actually an infinite number of halting probabilities, depending on your choice of programming language. After you choose a language, then you ask what the probability is that a program generated by coin tossing will eventually halt. And that gives you a different halting probability.

The numerical value will be different; the paradoxical properties are the same.

There are cases for which you can get a few of the first bits. For example, if Ω starts with 1s in binary or 9s in decimal, you can know those bits or digits, if Ω is $.11111\ldots$ base two or $.99999\ldots$ base ten. So you can get a finite number of bits, perhaps, of the numerical value, but if you have an N-bit formal axiomatic theory, then you can't get more than N bits of Ω. That's sort of the general result. It's irreducible logically and computationally. It's irreducible mathematical information.

That's the bad news. Algorithmic information theory (AIT) goes further than Turing, and picks out, from Turing's universal languages, maximally expressive programming languages—because those are the ones that you have to use to develop this theory where you get to Ω.

AIT has the notion of a maximally expressive programming language in which programs are maximally compact, and deals with a very basic complexity concept, which is the size of the smallest program to calculate something.

Now we have a better notion of perfection. Universal programming languages are not all equally good. We concentrate on a subset, comprising the ones that enable us to write the most concise programs. These are the most expressive languages, the ones with the smallest programs. This definition of complexity is a dry, technical way of expressing an idea in modern terms. But let me put this into medieval terminology, which is much more colorful. What we're asking is, how many yes/no decisions did God have to make to create something?—which is obviously a rather basic question to ask, if you consider that God is calculating the universe. I'm giving you a medieval perspective on these modern developments.

Theology is the fundamental physics, it's the theoretical physics of the Middle Ages.

The notion of the universal Turing machine that is used in AIT is Turing's very basic idea of a flexible machine. It's flexible hardware, which we call software. Now, AIT picks out a particular class of universal Turing machines U.

What are the universal computers U? A universal computer U has the property that, for any other computer C and its program p, the universal computer U will calculate the same result if you give it the original program p for C concatenated to a prefix π_C which depends only on the computer C that you want to simulate. π_C tells

U which computer to simulate. In symbols,

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

In other words, $\pi_C\, p$ is the concatenation of two pieces of information. It's a binary string. You take the original program p, which is also a binary string, and in front of it you put a prefix that tells you which computer to simulate. This means that these programs $\pi_C\, p$ for U are only a fixed number of bits larger than the programs p for any individual machine C. These U are the universal Turing machines that you use in AIT. These are the most expressive languages. These are the languages in which programs are as concise as possible. This is how you define program-size complexity. God will naturally use the most perfect, most powerful programming languages, when he creates the world, to build everything.

AIT is concerned with particularly efficient ways for U to be universal. Turing's original notion of universality was not this demanding. The fact that you can just add a fixed number of bits to a program for C to get one for U is not completely trivial. Let me tell you why. After you put π_C and p together, you have to know where the prefix ends and the program that is being simulated begins. There are many ways to do this. A very simple way to make the prefix π_C self-delimiting is to have it be a sequence of 0's followed by a 1:

$$\pi_C = 0^k1$$

And the number k of 0's tells us which machine C to simulate. That's a very wasteful way to indicate this.

The prefix π_C is actually an interpreter for the programming language C. AIT's universal languages U have the property that you give U an interpreter plus the program p in this other language C, and U will run the interpreter to see what p does.

If you think of this interpreter π_C as an arbitrary string of bits, one way to make it self-delimiting is to just double all the bits. 0 goes to 00, 1 goes to 11, and you put a pair of unequal bits 01 as punctuation at the end.

$$\pi_C : 0 \to 00, 1 \to 11, 01 \text{ at the end.}$$

This is a better way to have a self-delimiting prefix that you can concatenate with p. It only doubles the size, whereas the 0^k1 trick increases the size exponentially. And there are more efficient ways to make the prefix self-delimiting. For example, you can put the size of the prefix in front of the prefix. But it's sort of like Russian dolls, because if you put the size $|\pi_C|$ of π_C in front of π_C, $|\pi_C|$ also has to be self-delimiting:

$$U(||\pi_C||\, |\pi_C|\, \pi_C\, p) = C(p)$$

Anyway, picking U this way is the key idea in the original 1960s version of AIT that Andrey Kolmogorov, Ray Solomonoff, and I independently proposed. But ten years later I realized that this is not the right approach. You actually want the whole program $\pi_C\, p$ for U to be self-delimiting, not just the prefix π_C. You want the whole thing to be self-delimiting to get the right theory of program-size complexity.

Let me compare the 1960s version of AIT and the 1970s version of AIT. Let me compare these two different theories of program-size complexity.

In the 1960s version, an N-bit string will in general need an N-bit program, if it's irreducible, and most strings are algorithmically irreducible. Most N-bit strings need an N-bit program. These are the irreducible strings, the ones that have no pattern, no structure. Most N-bit strings need an N-bit program, because there aren't enough smaller programs. But in the 1970s version of AIT, you go from N bits to $N + \log_2 N$ bits, because you want to make the programs self-delimiting. An N-bit string will usually need an $N + \log_2 N$ bit program.

Actually, in 1970s AIT it's N plus $H(N)$, which is the size of the smallest self-delimiting program to calculate N, that is exactly what that logarithmic term is. In other words, in the 1970s version of AIT, the size of the smallest program for calculating an N-bit string is usually N bits plus the size in bits of the smallest self-delimiting program to calculate N, which is roughly

$$\log N + \log\log N + \log\log\log N + \ldots$$

bits long.

That's the Russian dolls aspect of this.

The 1970s version of AIT, which takes the idea of being self-delimiting from the prefix and applies it to the whole program, gives us even better perfect languages. AIT evolved in two stages. First we concentrate on those U with

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

with π_C self-delimiting, and then we insist that the whole thing $\pi_C\, p$ has also got to be self-delimiting. And when you do that, you get important new results, such as the sub-additivity of program-size complexity,

$$H(x, y) \leq H(x) + H(y),$$

which is not the case if you don't make everything self-delimiting. This just says that you can concatenate the smallest program for calculating x and the smallest program for calculating y to get a program for calculating x and y.

And you can't even define the halting probability Ω in 1960s AIT. If you allow all N-bit strings to be programs, then you cannot define the halting probability in a natural way, because the sum for defining the probability that a program will halt

$$\Omega = \sum_{p \text{ halts}} 2^{-(\text{size in bits of } p)}$$

diverges to infinity instead of being between zero and one. This is the key technical point in AIT.

I want the halting probability to be finite. The normal way of thinking about programs is that there are 2^N N-bit programs, and the natural way of defining the halting probability is that every N-bit program that halts contributes $1/2^N$ to the halting probability. The only problem is that for any fixed size N there are roughly on the order of 2^N programs that halt, so if you sum over all possible sizes, you get infinity, which is no good.

In order to get the halting probability to be between zero and one

$$0 \;<\; \Omega = \sum_{p \text{ halts}} 2^{-(\text{size in bits of } p)} \;<\; 1$$

you have to be sure that the total probability summed over all programs p is less than or equal to one. This happens automatically if we force p to be self-delimiting. How can we do this? Easy! Pretend that you are the universal computer U. As you read the program bit by bit, you have to be able to decide by yourself where the program ends, without any special punctuation, such as a blank, at the end of the program. This implies that no extension of a valid program is itself a valid program, and that the set of valid programs is what's called a prefix-free set. Then the fact that the sum that defines Ω must be between zero and one, is just a special case of what's called the Kraft inequality in Shannon information theory.

But this technical machinery isn't necessary. That $0 < \Omega < 1$ follows immediately from the fact that as you read the program bit by bit you are forced to decide where to stop without seeing any special punctuation. In other words, in 1960s AIT we were actually using a three-symbol alphabet for programs: 0, 1 and blank. The blank told us where a program ends. But that's a symbol that you're wasting, because you use it very little. As you all know, if you have a three-symbol alphabet, then the right way to use it is to use each symbol roughly one-third of the time. So if you really use only 0s and 1s, then you have to force the Turing machine to decide by itself where the program ends. You don't put a blank at the end to indicate that.

So programs go from N bits in size to $N+\log_2 N$ bits, because you've got to indicate in each program how big it is. On the other hand, you can just take subroutines and concatenate them to make a bigger program, so program-size complexity becomes subadditive. You run the universal machine U to calculate the first object x, and then you run it again to calculate the second object y, and then you've got x and y, and so

$$H(x,y) \leq H(x) + H(y).$$

These self-delimiting binary languages are the ones that the study of program-size complexity has led us to discriminate as the ideal languages, the most perfect languages. We got to them in two stages, 1960s AIT and 1970s AIT. These are languages for computation, for expressing algorithms, not for mathematical reasoning. They are universal programming languages that are maximally expressive, maximally concise. We already knew how to do that in the 1960s, but in the 1970s we realized that programs should be self-delimiting, which made it possible to define the halting probability Ω.

That's the story, and now maybe I should summarize all of this, this saga of the quest for the perfect language. As I said, the search for the perfect language has some negative conclusions and some positive conclusions.

Hilbert wanted to find a perfect language giving all of mathematical truth, all mathematical knowledge; he wanted a formal axiomatic theory for all of mathematics. This was supposed to be a Theory of Everything for the world of pure mathematics. And this cannot succeed, because we know that every formal axiomatic theory is incomplete, as shown by Gödel, by Turing, and by me. Instead of finding a perfect language, a perfect formal axiomatic theory, we found incompleteness, uncomputability, and even algorithmic irreducibility and algorithmic randomness.

That's the negative side of this story, which is fascinating from an epistemological point of view, because we found limits to what we can know; we found limits of formal reasoning.

Now interestingly enough, the mathematical community couldn't care less. They still want absolute truth! They still believe in absolute truth, and that mathematics

gives absolute truth. And if you want a proof of this, just go to the December 2008 issue of the *Notices of the American Mathematical Society*. That's a special issue of the *Notices* devoted to formal proof.

The technology has been developed to the point where they can run real mathematics, real proofs, through proof-checkers, and get them checked. A mathematician writes the proof out in a formal language, and fills in the missing steps and makes corrections until the proof-checker can understand the whole thing and verify that it is correct. And these proof-checkers are getting smarter and smarter, so that more and more of the details can be left out. As the technology improves, the job of formalizing a proof becomes easier and easier. The formal-proof extremists are saying that in the future all mathematics will have to be written out formally and verified by proof-checkers.

The position of these extremists is that in the future all mathematics will have to be written out in a formal language, and you will have to get it checked before submitting a paper to a human referee, who will then only have to decide if the proof is worth publishing, not whether the proof is correct. And they want a repository of all mathematical knowledge, which would be a database of checked formal proofs of theorems.

I'm not disparaging this extremely interesting work, but I am saying that there's a wonderful intellectual tension between it and the incompleteness results that I've discussed in this talk. There's a wonderful intellectual tension between incompleteness and the fact that people still believe in formal proof and absolute truth. People still want to go ahead and carry out Hilbert's program and actually formalize everything, just as if Gödel and Turing had never happened!

I think this is an extremely interesting and, at least for me, a quite unexpected development.

These were the negative conclusions from this saga. Now I want to wrap this talk up by summarizing the positive conclusions.

There are perfect languages for computing, not for reasoning. They're computer programming languages. And we have universal Turing machines and universal programming languages, and although languages for reasoning cannot be complete, these universal programming languages are complete. Furthermore, AIT has picked out the most expressive programming languages, the ones that are particularly good to use for a theory of program-size complexity.

So there is a substantial practical spinoff. Furthermore, since I've worked most of my professional career on AIT, I view AIT as a substantial contribution to the search for the perfect language, because it gives us a measure of expressive power, and of conceptual complexity and the complexity of ideas. Remember, I said that from the perspective of the Middle Ages, that's how many yes/no decisions God had to make to create something, which obviously he will do in an optimal manner.

From the theoretical side, however, this quest was disappointing: due to Gödel incompleteness and because there is no Theory of Everything for pure mathematics. In fact, if you look at the bits of the halting probability Ω, they show that pure mathematics contains infinite irreducible complexity, and in this precise sense it is more like biology, the domain of the complex, than like theoretical physics, where there is still hope of finding a simple, elegant TOE.

So this is the negative side of the story, unless you're a biologist. The positive side is we get this marvelous programming technology. So this dream, the search for the

perfect language and for absolute knowledge, ended in the bowels of a computer, it ended in a Golem.

How would all this look to someone from the Middle Ages? This quest, the search for the perfect language, was an attempt to obtain magical, God-like powers.

Let's bring someone from the 1200s here and show them a notebook computer. You have this dead machine, it's a machine, it's a physical object, and when you put software into it, all of a sudden it comes to life!

So from the perspective of the Middle Ages, I would say that the perfect languages that we've found have given us some magical, God-like power, which is that we can breathe life into some inanimate matter. Observe that hardware is analogous to the body, and software is analogous to the soul, and when you put software into a computer, this inanimate object comes to life and creates virtual worlds.

So from the perspective of somebody from the year 1200, the search for the perfect language has been successful and has given us some magical, God-like abilities, except that we take them entirely for granted.

Thanks very much!

Chapter 6

A Life in Mathematics (2021)

Gregory Chaitin's life in mathematics punctuated by some photographs taken during crucial episodes in his career.

New York, 1947–1965, Buenos Aires, 1966–1975

I was born in 1947 in Chicago and grew up in Manhattan, surrounded by books from the Museum of Modern Art (MoMA) and by issues of *Farm Journal*, *Theatre Arts Magazine* and *Scientific American*. My parents were involved with the theatre and with the United Nations. I practically lived in the Donnell branch of the New York City Public Library on 53rd street, in the MoMA across the street, and in Central Park a block away from our home between 68th and 69th street on Madison Avenue (819 Madison Ave. to be precise). I studied in P.S. 6, in the Bronx High School of Science, and in the Columbia University Science Honors Program for bright high school students, where I learned to program and was given the run of the Columbia University libraries.

Before leaving for Argentina I was briefly at the City College of the City University of New York, where I was excused from attending classes to write my first papers on program-size complexity and defining randomness, which would subsequently be published in the *Journal of the ACM*. The editor of the *Journal of the ACM* to whom I submitted these papers was Prof Martin Davis, who had been a student of Emil Post and who is known for his work on Hilbert's 10th problem.

Furthermore, although I never graduated from City College because I left for Argentina, I was awarded the Nehemiah Gitelson award "for the student who in his undergraduate career best exemplifies the spirit of the search for truth," and the Belden Mathematical Prize (gold medal), both of which are usually only given to graduating students. And an article about me appeared in the *New York Times* containing praise by Prof Gian-Carlo Rota of MIT, who had been a student of Jack Schwartz (see below).

From 1966 through 1975 I lived in Buenos Aires, Argentina (BA), where my parents were born and where I enjoyed rowing in the Tigre river delta and attending ballet and light opera at the Teatro Colon opera house, not to mention the European style cafés and restaurants, and where I joined IBM before being transferred to the Watson Research Center in New York in 1975.

My return to New York was the result of a chance meeting in BA with the dis-

tinguished Courant Institute of New York University mathematician Prof Jacob T. ("Jack") Schwartz, who was impressed by my very simple proof that an N-bit formal axiomatic theory cannot provide individual examples of provably "elegant" programs greater than $N + c$ bits in size, an "elegant" program being one with the property that no smaller program can produce the same output that it does.

By the way, Jack Schwartz, like Martin Davis, had at City College been a student of Emil Post, who discovered incompleteness and uncomputability independently of Gödel and Turing, and who subsequently went deeper into these topics than either of these two more famous men, who had many other interests.

While living in Buenos Aires, I published five papers in the *Journal of the ACM* and two papers in the *IEEE Transactions on Information Theory*, and discovered information-theoretic incompleteness during a visit to Rio de Janeiro in 1970 and the halting probability during a visit to New York in 1974.

Visit to Rio de Janeiro, 1970

The photo on this page was taken while I was visiting Rio de Janeiro in 1970. That is where I wrote my first paper on information-theoretic incompleteness, a Pontificia Universidade Católica (PUC) research report. I am dancing with the "porta-bandeira" (flag bearer) of the Salgueiro samba school before she enters the Carnival parade, which was then on Rio Branco avenue. This photo was taken in the vicinity of the Candelária church where the samba schools were assembling and waiting for their turn to parade. At that time I was twenty two, belonged to a rowing club in the Tigre river delta in Buenos Aires, and was very fit. This photograph is courtesy of the German numerical analyst Peter Albrecht, who was visiting Rio at the time.

Carnival in Rio

The PUC research report containing my first result on information-theoretic incompleteness actually consisted of two parts. The first part was called "Computational Complexity and Gödel's Incompleteness Theorem," and the second part was called "To a Mathematical Definition of 'Life'." It marked the beginning of a lifelong attempt to find the mathematical basis of biology, which was to culminate decades later in my book *Proving Darwin: Making Biology Mathematical* (2012).

And in Rio I was fortunately able to purchase a copy of the *LISP 1.5 Programmer's Manual* (MIT Press, 1962), thus beginning a lifelong infatuation with LISP and with inventing LISP dialects, programming LISP interpreters, and proving theorems about LISP program-size complexity.

Visit to New York, 1974

In the first few months of 1974 I traveled from Buenos Aires to New York as a "summer visitor" at the IBM T. J. Watson Research Center in Yorktown Heights. I lived in the White Plains YMCA—where I would swim—and commuted to the Watson Center by train and taxi.

It was during this visit that I discovered or invented the halting probability Ω. I remember the exact moment. I had been invited to give a lecture at a university somewhere in the United States—every week it was a different one—and was flying back to New York. At the precise moment that I realized that the halting probability was irreducible or algorithmically random, I was looking out the window and saw an unmistakable sight, the Pentagon in Washington, DC.

In Terry Fine's office at Cornell University

Due to the usual delays for refereeing and such, the halting probability did not appear in print until the next year, 1975, in my fifth *Journal of the ACM* paper, "A Theory of Program Size Formally Identical to Information Theory."

By the way, the halting probability was originally ω, but the set theorist Robert Solovay, who was visiting the Watson Research Center, suggested to me that Ω might be better because in set theory ω stood for the set of natural numbers $\{0, 1, 2, 3, \ldots\}$.

During this visit to the Watson Research Center I also corrected the proofs of one of my first publications on incompleteness, destined to appear later in the year, an invited

paper "Information-Theoretic Computational Complexity" in the *IEEE Transactions on Information Theory*, with an appendix giving the mathematical details, which proofs I was to send to Gödel, as I will tell below.[1]

And I had two very interesting experiences.

The first was that I attended a lecture at the New York Academy of Sciences in Manhattan by a mathematician I admired, Mark Kac. The lecture was on randomness, and Kac's thesis was that randomness was an interesting but slippery notion that resisted precise definition. He concluded his lecture with the following words: "In spite of this, a definition of randomness has been proposed by Kolmogorov and by a young fellow in Argentina, Gregory Chaitin." I stood up and said, "No, I'm here now!" Pandemonium, over which Kac declared, "This was not rehearsed!"

After the talk a gentleman came up to me and said, "I'm Dennis Flanagan, the Editor of *Scientific American*." And he told me the following story about Gödel. At the time Flanagan was living in Princeton, New Jersey, and he had just published a wonderful article, "Gödel's Proof" by Ernest Nagel and James R. Newman (1956), later expanded into a small book (NYU Press, 1958) that completely obsessed me from the moment it appeared in the New York City public library (at that time I lived in Manhattan). Gödel was not known to the general intellectual public yet—that article and that book were to change that—and few people had seen a photo of Gödel and knew how he looked. However, Flanagan had sent the well-known portrait photographer Arnold Newman to Princeton in order to be able to include an image of Gödel in the article about him in *Scientific American*, resulting in a stark portrait of an angry-to-be-disturbed Gödel sitting in front of a bare blackboard that has been reproduced many times.

So Flanagan knew how Gödel looked. And one hot, humid summer day Flanagan was walking down the street in Princeton, a small town, and saw Gödel approaching. He prepared to introduce himself as the publisher of the article about Gödel's proof. At that moment, however, a scantily clad beautiful young female student (we used to call them "co-eds" from the word "co-education") passed by, and Gödel stopped dead in his tracks to admire her. As they say in French, "La belle opportunité est perdu!" Flanagan did not dare to interrupt Gödel!

The second amazing experience was that I somehow managed to make a phone call to Gödel's office at the Princeton Institute for Advanced Study (IAS), a cold call as they say in the world of sales, and Gödel himself picked up the phone. "Professor Gödel," I said, "I am extremely fascinated [obsessed would have been more accurate] by your incompleteness theorem, and I have a new proof based on the Berry paradox instead of the Epimenides paradox [the paradox of the liar, 'This statement is false']." He replied, "It doesn't matter which paradox you use!" In fact, he says this in the introduction to his famous 1931 paper, which I was familiar with. So I was prepared, and I immediately answered, "Yes of course, but this suggests to me a new information-theoretic view of incompleteness, which I would very much like to visit you and tell you about." He replied, "Send me a paper of yours on this subject, and I will take a look at it and decide if I give you an appointment." So I sent him the proofs of my

[1]However, my best paper on incompleteness was probably "Gödel's Theorem and Information" published in the *International Journal of Theoretical Physics* years later, in 1982, and then reprinted in Tymoczko, *New Directions in the Philosophy of Mathematics* (Princeton University Press, 1998), together with the paper that I sent to Gödel.

as-yet-unpublished 1974 IEEE paper. Then I called him back, and he commented "Very interesting, your complexity measure is an absolute notion [like computability as contrasted with provability, which depends on the axioms]." And he gave me an appointment!

The great day arrived, and I had already figured out how to take the train from Yorktown Heights into New York City and from there to Princeton, New Jersey, and how long that would take. It was the week before Easter, and that weekend I was supposed to leave NY and fly back to Buenos Aires. There had been a Spring snowstorm, nothing serious, nothing that would stop me from visiting my hero, Kurt Gödel. I was about to leave my office at IBM for the train station, when the phone rang, and a voice, a terrible voice, that of Gödel's secretary, announced that Gödel was very careful about his health and because it had snowed he was not coming into his office that day and therefore my appointment was canceled!

So this is how I spoke to Gödel on the phone twice but never met him. In retrospect, I think this is a much more interesting story than if I had actually met Gödel. It illustrates the surreal quality of interactions with Gödel.

The next week I stopped on my way back to Buenos Aires to present "A Theory of Program Size Formally Identical to Information Theory" at Stanford University.

However, the *annus mirabilis* 1974 was not yet over. Back in Buenos Aires, I was summoned by the head of IBM Argentina, Mr Benito Esmerode. The moment I sat down in Mr Esmerode's office, the phone rang. It was the head of IBM, Thomas J. Watson Jr. "Yes," said Mr Esmerode, "he is here in my office now, and yes, of course we will pay for his trip to the University of Notre Dame!"

What had happened? The IEEE was holding their 1974 International Symposium on Information Theory later that year at Notre Dame University, and the organizers wanted me to present "A Theory of Program Size Formally Identical to Information Theory" in their opening plenary session. But I had told them I couldn't travel to Indiana. So the president of Notre Dame wrote to Thomas J. Watson Jr. and asked for his help. Problem solved.

That was my second trip from Buenos Aires to the United States in 1974. I was transferred from IBM Argentina to the Watson Research Center in 1975, the year that my article on "Randomness and Mathematical Proof" appeared in *Scientific American*.[2]

Years later my friend Cristian Calude from the University of Auckland was visiting me at the Watson Research Center, and we decided to make a pilgrimage to Princeton. We found Einstein's former home near the IAS, Gödel's former home in a much poorer part of town, and Gödel's and John von Neumann's graves in the Princeton Cemetery. Einstein is not there. He was cremated and his ashes were scattered at an undisclosed location, as he had wished.

Furthermore, as we stood looking at Gödel's home, the couple who were renting it from the current owner came out and invited us in. It turns out that much remained exactly as it had been when Kurt and his wife Adele lived there, in particular the heavy sound-proofing so that Gödel could work undisturbed in his study, and a shrine to the Virgin Mary in the garden, but not Adele's infamous pink flamingo, which Gödel found "charming."

[2]To be followed by "Randomness in Arithmetic" in 1988 and by "The Limits of Reason" in 2006.

With Cris Calude at Gödel's grave in Princeton, New Jersey

Interregnum, 1976–1986

Now there is a gap, during which I worked mostly on practical hardware and software engineering for IBM, which was a lot of fun. It was great to learn about hardware design and architecture issues, and to work on compilers and on some operating system components. I was part of a small, really terrific team, which was very educational.

After all, what do you do when you don't have a great new mathematical idea? I like to turn to practical engineering problems, which give one a feeling of accomplishment, while waiting for lightening to strike again.

We worked the way Elon Musk works at SpaceX, quickly lashing prototypes together, using them ourselves, and improving them as we went along. Theory was on a backburner.

This period of my career has been described in a perceptive essay by my former colleague, Rocky Bernstein, in his essay "Greg Chaitin, Computer Programmer" at http://rocky.github.io/gjchaitin.pdf.

In particular, I remember four projects.

The first was a timer for a proposed new processor design. It was a register-level simulator that we would run an instruction stream through to see how well the processor performed.

The second project I remember did register allocation via graph coloring for the back-end of an optimising compiler.

The third project was a binder that separated data and code, that attempted to re-order everything to minimize the working set, and that featured a garbage collector so that an entire run-time library could be bound with the output of the compiler and the unnecessary run-time routines would disappear.

Finally a project that was a hobby, to teach myself theoretical physics by programming simulations using the fundamental equations of theoretical physics. This included: (a) calculating and graphing the trajectory of a satellite orbiting a point mass according to Newtonian physics and according to Einstein's theory of general relativity; (b) a numerical verification of Einstein's field equations at a point near the event horizon

of a Schwarzschild black hole; (c) showing the propagation of an electromagnetic wave according to the traditional formulation of Maxwell's equations and according to the more sophisticated vector potential formulation; (d) showing Schrödinger's equation in action by scattering the psi function probability wave for a solitary electron against differently shaped one-dimensional potentials; and finally (e) attempting to repeat the same calculations, as much as possible, using the Feynman path integral, sum over all histories reformulation of quantum mechanics.

To get these physics working models to function properly, I had to learn enough numerical analysis to do numerical solutions of partial differential equations. I fortunately could take advantage of the expertise of a member of the theoretical physics group who had learned numerical analysis simulating atom and hydrogen bomb explosions at one of the United States government national labs, Gordon Lasher, a very nice guy.

This then became my "Computer Gallery of Mathematical Physics" course, the idea being to teach fundamental physics by showing how to do the calculations instead of presenting the mathematics in the traditional way. Finally it morphed into my "APL2 Gallery of Mathematical Physics" course that took advantage of the fact that APL2 was an executable notation as concise as the equations of mathematical physics, but, it must be confessed, much more cryptic, because it employed so many special characters as one-character built-in mathematical functions (such as matrix multiply and matrix inversion). This APL2 Gallery also earned me my first trip to Switzerland and my first trip to Japan, an unexpected but most welcome bonus. And it was so concise that all the APL2 code for the course eventually ended up hanging, beautifully framed, as a single piece of conceptual art on the wall of our apartment in Rio de Janeiro.

Of course, now the right way to do all this would be to reprogram it in Stephen Wolfram's Mathematica, or, should I say, in the Wolfram Language.

As I said before, theory was on a backburner. I did of course accept invitations to lecture and to write some papers, but there was no fundamental new mathematical idea during this period. However, my understanding of incompleteness and its philosophical implications slowly advanced, helped by Martin Davis' suggestion that I take a look at Gödel's essay "What is Cantor's Continuum Problem?" The result was my best paper on incompleteness, "Gödel's Theorem and Information," published in the *International Journal of Theoretical Physics* in 1982, and then reprinted in Tymoczko, *New Directions in the Philosophy of Mathematics* (Princeton University Press, 1998), a collection of papers supporting a "quasi-empirical" view of math.

Visit to Vienna, 1991

On the next page I am in the small classroom at the University of Vienna where Gödel taught. The two lectures that I gave in Vienna in January 1991—one in this room and one at the Technical University of Vienna—are contained in my 1992 book *Information-Theoretic Incompleteness* (World Scientific). My 2002 book *Conversations with a Mathematician* (Springer-Verlag) also includes the Technical University lecture and provides additional background information.

I was invited to Vienna because of my 1987 Cambridge University Press monograph *Algorithmic Information Theory*, the subject of a highly visible and very favorable review in the prestigious journal *Nature* ("The Ultimate in Undecidability," 1988).

In Gödel's classroom: Hier Wirkte Kurt Gödel von 1932–1938

And I was greeted in Vienna by a full-page article in the newspaper *Der Standard* entitled "Gödeliger aus Gödel," Out-Gödeling Gödel.

I think of this as my "Randomness in Arithmetic" episode, which was the title of the *Scientific American* article that I published in 1988 summarizing the results in my 1987 Cambridge book. With it I got out of my system the fascination with Hilbert's 10th problem that I inherited from Martin Davis and, actually, from Emil Post, who had told Martin that Hilbert's 10th problem "begged for an undecidability proof."

After this "Randomness in Arithmetic" episode, I spent years programming out algorithmic information theory in LISP for my three Springer-Verlag textbooks,[3] and further years absorbed in studying Leibniz, who had also fascinated Gödel.

And then I attempted to write the kind of book that as an adolescent had inspired me to become a mathematician. The result was *Meta Math!: The Quest for Omega* (2005), which I summarized in my 2006 *Scientific American* article on "The Limits of Reason." In both of these publications Leibniz figures prominently.

Why did I embark upon a serious study of Leibniz? In fact, it was completely fortuitous. I had long been aware of a reference in one of Hermann Weyl's works on the philosophy of science to Leibniz's reflections on complexity in the *Discours de métaphysique* (1686). But I did not follow this up until an invitation arrived for me to give a talk at a September 2002 meeting of the German Philosophical Society in Bonn. So began my Leibniz phase, which continued until the tricentennial of Leibniz's death in 2016, when I give lectures on Leibniz in Turin, Kraków and Singapore. Furthermore, Ugo Pagallo, who had just published *Leibniz—Una breve biografia intellettuale*, happened to visit Brazil in 2016, so my wife and I organized a Paquetá island Leibnizfest

[3] *The Limits of Mathematics* (1998), *The Unknowable* (1999), and *Exploring Randomness* (2001).

with Ugo and friends at our weekend home there and at a fine restaurant on the island at the Casa de Artes.

I must confess that it had not been easy for me to study Leibniz. I amassed a considerable collection of Leibniz books, a number of them out of print and hard to obtain. But it was all worth it, and was reflected not only in my talks but also in a series of papers such as "Leibniz, complexity, and incompleteness," which I published in the *APA Newsletter on Philosophy and Computers* in 2009.

I should say more about the genesis of *Algorithmic Information Theory*, my first book. In 1986 I received a letter from Cambridge University Press. They informed me that they were launching a new series, *Cambridge Tracts in Theoretical Computer Science*, with the goal of showing that there is intellectually significant theory behind computer science. And to make this point as clearly as possible, they were asking me to write the first book in their series!

I took this letter to Ralph Gomory, at that time the head of research. He had started his career as a mathematician, and I admired him and felt that he could have been minister of science and technology for a nation. He had given me a helping hand several times, at crucial junctions in my career at IBM. I was supposed to be working on practical problems, but Ralph picked up the phone, called the head of the physics department, and I was given a one-year internal sabbatical to write my book!

And I am proud that Jack Schwartz wrote the foreword for *Algorithmic Information Theory*. Jack was a spectacular mathematician, and a workaholic. He had studied with Emil Post, who had recognized his talent, and he was the co-author (some people said practically the sole author) of Dunford and Schwartz, *Linear Operators*, an exhaustive and massive three–volume masterpiece, the likes of which we shall probably never see again.

To indicate the kind of man that Jack was, I will repeat a story that his student Gian-Carlo Rota tells in the *AMS Notices*. There is a certain important theorem that appears in Dunford and Schwartz that is usually credited to a well-known mathematician. Gian-Carlo was, however, surprised that in Dunford and Schwartz it was not so credited. Then he noticed that Dunford and Schwartz was published before that well-known mathematician's paper! Jack had proved the theorem, but had not bothered to claim credit. For Jack, what counted was doing the mathematics, not who got the credit. I have tried as best I could to follow his example.

Greg and Virginia in New York, 2008

The photo on the next page shows us just before we got married in August, 2008. We are on Bear Mountain, and in the background you can see the Bear Mountain Bridge and the Hudson River—actually a glacially-cut fjord with excellent hiking trails. This photo was taken by Karol Jałochowski for *Polityka* magazine, who later filmed the *Against Method* documentary—on creativity in mathematics and in biology—at our Paquetá island weekend home near Rio de Janeiro.[4]

The week before we got married I was reading David Berlinski's *The Devil's Delusion: Atheism and Its Scientific Pretensions* (2008), which has a critique of Darwinism, and I had the idea of trying to model evolution as a random walk in software space

[4]Available at https://youtu.be/uEtqJSRz3mE.

At Bear Mountain Bridge

("metabiology"). In *Proving Darwin: Making Biology Mathematical* (2012), I credit Virginia as my muse, and I refer to metabiology as "our child," as indeed it was at that time because our two children had not yet been born, and in fact the prospect of having children, although desired by both of us, unfortunately seemed rather remote.

Jack Schwartz also played a role. In our numerous dinners in hole-in-the-wall ethnic restaurants in Greenwich village in Manhattan, he would talk to me about molecular biology, which he was studying with his usual intensity, thoroughness and omnivorous intellectual appetite. "DNA," he said, "is just a digital programming language!" He encouraged me not to give up and to continue thinking about biology. Unfortunately he passed away in 2009, so I was never able to show him *Proving Darwin*.

I had been searching for this idea—random walk in software space!—my entire life, but I was only able to come up with my mathematical theory of biology after retiring from IBM and moving to Brazil. And it was also in Brazil that Virginia and I finally resorted to *in vitro* fertilisation treatments, becoming parents later in life than is usually the case. That story will be told in Virginia's forthcoming book *Maternidade Tardia: Contextos e Caminhos*.

Strangely enough, all the technical tools needed for metabiology were already in my 1975 paper "A Theory of Program Size Formally Identical to Information Theory," but in 1975 I lacked the necessary sophistication.[5]

Nor did I have Virginia to provide a philosophical perspective on several crucial issues:[6] (a) the oracle in metabiology corresponds to the environment; (b) the global algorithmic mutations used in metabiology instead of SNPs and indels are a key con-

[5]It took me many years to understand what mathematical methods could or could not achieve, and to free myself from the formulation in John von Neumann's posthumous notes, *Theory of Self-Reproducing Automata*, edited and completed by Arthur W. Burks (University of Illinois Press, 1966).

[6]See Virginia's paper "Metabiology, Interdisciplinarity and the Human Self-Image" in Wuppuluri, Doria, *Unravelling Complexity: The Life and Work of Gregory Chaitin* (2020).

tribution, not an embarrassment, and in fact have been picked up by Hector Zenil and his collaborators[7] and may explain the major transitions in evolution; and (c) metabiology, in emphasizing open-ended creativity instead of adaptation to the environment and selfish genes, goes beyond Darwinism as it is currently understood—and provides us with a different and more flattering human self-image.

With Maria Clara and João Bernardo in Paquetá, 2021

Coda

I offer this story of a life in math as food for thought. Why is one period of a person's life more fertile than another? Why is one particular decade in the life of a nation or of a city more creative? Where do new ideas come from? What kind of stimulus is necessary? Why did it take me so many years to come up with a simple model of evolution?[8]

Nations, individuals and corporations seem to have periods of youthful vigor and periods of decline. Maybe I am fooling myself, but the postwar United States in the

[7]Please see the article in *Quanta* magazine at https://www.quantamagazine.org/computer-science-and-biology-explore-algorithmic-evolution-20181129/

[8]Fortunately for my self-esteem, Jacques Hadamard, in his book *The Psychology of Invention in the Mathematical Field* (Princeton University Press, 1945), gives many examples of mathematicians, including himself, who failed to see more or less obvious consequences of their own work. I believe this is sometimes referred to as "tunnel vision."

1950s seemed particularly optimistic and energetic—as evidenced by the large families and the thick issues of *Scientific American*—a period captured in my father Norman Chaitin's 1962 feature film *The Small Hours*—preserved in the MoMA film library—an Argentine intellectual's depiction of struggling artist types in postwar Manhattan.

Corporations also go from dynamic, energetic, can-do beginnings into bureaucratic decline.

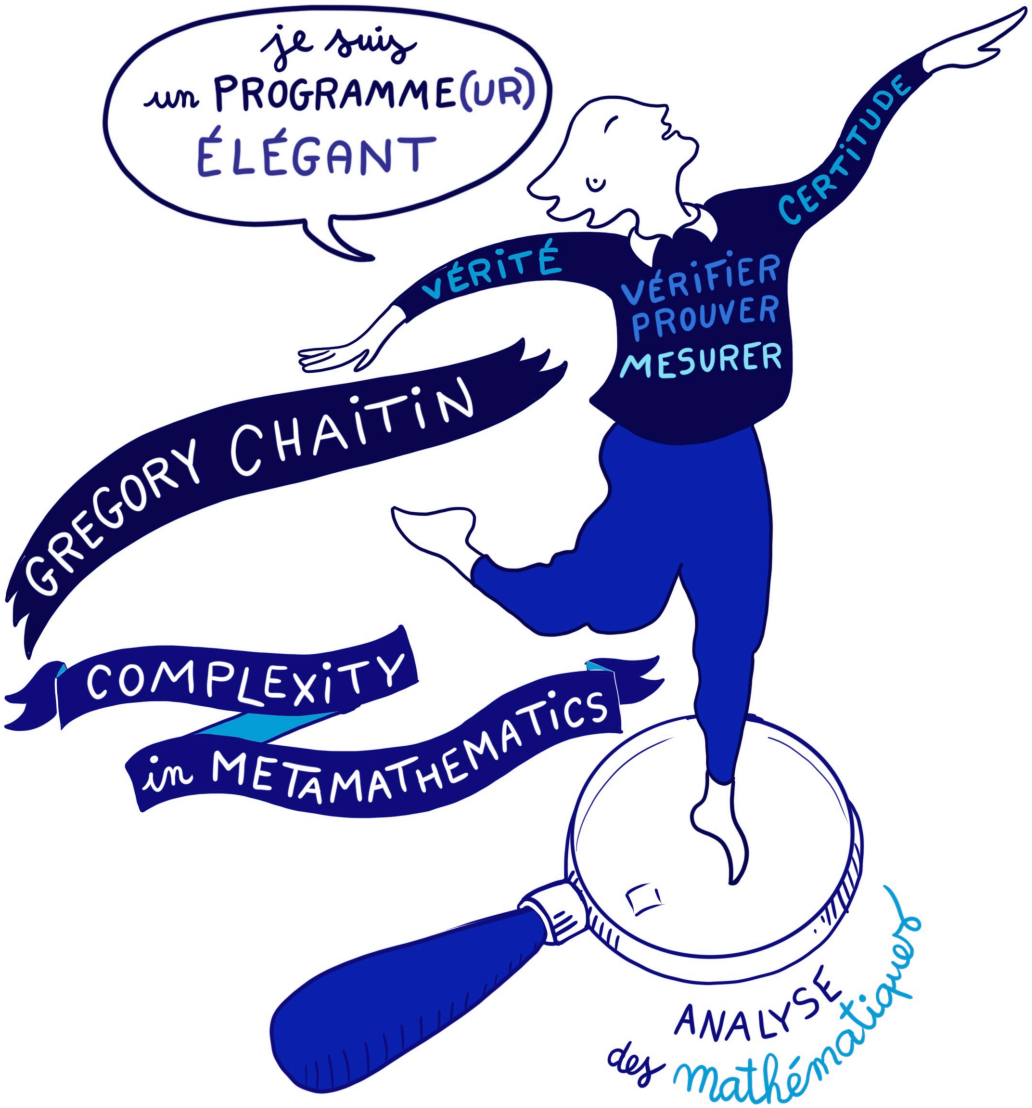
The mystery of creation is evident in my little story, but even more so in the lives of Leonhard Euler and Srinivasa Ramanujan, two spectacular cases of overflowing creativity, of overabundant gifts. And then there are mathematicians like John von Neumann or Jack Schwartz—or even my friend the physicist Stephen Wolfram—who seem to be mutant life forms, aliens more or less capable of pretending to be human beings. How do such minds function? How is the human spirit capable of such achievements? Will we be able to create artificial intelligences at that level?

Gregory Chaitin is an Argentine-American mathematician living in Rio de Janeiro, and a lifetime honorary professor of the University of Buenos Aires with an honorary doctorate in philosophy from the University of Córdoba, the oldest university in Argentina and one of the oldest in South America. He was formerly at the IBM Watson Research Center in New York, where he was part of a small team that developed the Power processor architecture and its associated software.

On the theoretical side, Chaitin is best known for his discovery of the remarkable Ω number, a concrete example of irreducible complexity in pure mathematics, and which shows that mathematics is infinitely complex. For this he was awarded the Leibniz Medallion by Wolfram Research in 2007. He has also proposed modeling evolution as a random walk in software space ("metabiology").

Among his books are: *Algorithmic Information Theory*; *Conversations with a Mathematician*; *Meta Math!*; and *Proving Darwin*.

Festschriften: Cristian S. Calude, *Randomness and Complexity, from Leibniz to Chaitin*, World Scientific, Singapore, 2007; Gregory Chaitin, *Thinking about Gödel and Turing: Essays on Complexity, 1970–2007*, World Scientific, Singapore, 2007; Shyam Wuppuluri, Francisco Antonio Doria, *Unravelling Complexity: The Life and Work of Gregory Chaitin*, World Scientific, Singapore, 2020.


je suis un PROGRAMME(UR) ÉLÉGANT
VÉRITÉ
VÉRIFIER PROUVER MESURER
CERTITUDE
GREGORY CHAITIN
COMPLEXITY in METAMATHEMATICS
ANALYSE des mathématiques