Part One: Inside Our Computable World, and the Mathematics of Universality

Turing-Welchman Bombe rebuild at Bletchley Park. Reproduced with permission of Jochen Viehoff.

In 1954, Turing's last published paper, an article in *Penguin Science News*, conveyed the significance of computability in mathematics to a wide audience. But he had done little to promote his own formalism of Turing machines: just one paper applying it to a decidability question in algebra and nothing at all for emergent computer science. This left a gap in mathematical logic that was filled systematically by **Martin Davis** in his book *Computability and Unsolvability* from 1958. Now Davis emulates Turing's 1954 semi-popular work in explaining the question of Hilbert's Tenth Problem about the solvability of Diophantine equations, which Turing's definition of computability had rendered completely well defined. Martin Davis himself made a major contribution towards its beautiful resolution in 1970, and here he points to how this work refined and extended key aspects of Turing's seminal 1936 paper.

The most striking of Martin's observations relate to *universality* and *incomputability*. The results he describes as arising from the celebrated work of him and

his co-solvers elegantly clarify the reach and limitations of Turing computability at the classical level. Turing's later work anticipated today's interest in 'new computational paradigms' derived from processes in Nature, and the scope of *virtual machines*. There are important and deep questions concerning the extent to which these new models of computation are subsumed within Turing's 1936 paradigm – at a practical level via the development of virtual machines. Notions of universality and reach continue to raise basic questions for today's researchers.

Martin Hyland fills in a missing piece of history in a way that evokes and explores Alan Turing's close relationship with Robin Gandy. This is a counterpart to Alan Garner's story: another hidden history, in the same time-frame. He brings to life a topic that Turing studied but never brought to fruition: the theory of types, then totally abstract but which now would be seen as vital to formal structures in computer science (a concept that in Turing's time did not exist) and, more widely, in relation to the use of language in scientific theories.

Martin's contribution focuses on little-known ideas quite fundamental to a number of contemporary scientific preoccupations. The title of Robin Gandy's early research thesis, 'Some Considerations concerning the Logical Structure underlying Fundamental Theories in Theoretical Physics', provides a clear link to subsequent convergences between science and logical frameworks. A key aspect of later developments is the interrelating interests of Turing and Gandy in Church's theory of types. Martin notes that:

> Turing's influence as Gandy's supervisor relates specifically to Type Theory, and I started this paper with the thought that Turing's interest in the area is largely forgotten.

He proclaims "It is time to say something about that interest." What follows is one of the most fascinating discussions in this whole volume, and brings out Turing's distinctive, and often prescient, linking of the abstract and the concrete in a quite unique way. Nobody else in the post-centenary period has covered this ground, and certainly not in such an authoritative and informative way.

Andrew Booker explains Turing's work in analytic number theory, a story which brings in the special machine Turing started to build in 1939 and then his programming of the Manchester computer in 1950, thus making an explicit example of how software would replace the engineering of special machines. Booker's discussion leads into the modern status of the outstanding problem of the Riemann hypothesis and the lasting significance of Turing's computational ideas.

Andrew Booker's description of the work in number theory emphasizes the extent to which Turing's interests tend to arise from more far-reaching questions. In this case there is an awareness of how the advent of the computer will inevitably impact on the very nature of proof in mathematics and a prescient anticipation of how

mathematicians will adapt to the changes. A theme familiar from later writings of Turing is speculation about how computers will complement human creativity and about the evolving nature of the balance between mind and machine. As Booker notes, there may even be ethical consequences of this increasingly important and complex relationship.

Ueli Maurer takes up a parallel story, inspired by the little we know of Turing's pre-war ideas in cryptology and the emergent concept of crypto security. He sets out the terrain in the following terms:

> Computation and information are the two most fundamental concepts in computer science, much like mass, energy, time, and space are fundamental concepts in physics.

He goes on to describe the seminal and complementary roles played by Alan Turing's computational model and Claude Shannon's information theory. Beyond the practicalities of modern cryptography, his discussion relates to the status of another huge unsolved modern problem, that of the P = ?NP question. Ueli Maurer writes

> One can only speculate what Turing might have been able to achieve in the field of theoretical cryptography had he spent more time on the subject.

But we do not know what he may have done after 1938 that remains secret. The Delilah speech-encipherment project reports remained secret for over 50 years, and the full description was published only in 2012. Also in 2012, two quite fundamental Turing papers were released from secrecy, explaining the basis of Bayesian analysis and its application to the Naval Enigma 'Banburismus' method. There may well be much more to follow.

The timely releases by GCHQ in honour of the Turing centenary connect us to one of Alan Turing's most original and important contributions at Bletchley Park. Essential to the bringing of intercepted German messages within the scope of the Bombe and Colossus decoding machines was the application of what were recognisably Bayesian statistical techniques. In 'Alan Turing and Enigmatic Statistics', statistician **Kanti Mardia** and logician **Barry Cooper** take a closer look at the history at Bletchley Park, and at the current significance of the statistics for today's 'big data'. The "enigmatic statistics", we read, "foreshadowed the style of what is now called Statistical Bioinformatics".

Following informative examples, and an outline of the content and significance of the released papers, Mardia and Cooper return us to the mathematics of typed information and the use of sampling techniques for 'type reduction' to data accessible to classical Turing computation. From this perspective, they describe how the cryptanalytical role of Turing's Bayesian techniques potentially take us to a better understanding of the challenge of 'big data' in a wider context.

1

Algorithms, Equations, and Logic

Martin Davis

From the beginning of recorded history, people have worked with numbers using *algorithms*. Algorithms are processes that can be carried out by following a set of instructions in a completely mechanical manner without any need for creative thought. Until the 1930s no need was felt for a precise mathematical definition or characterization of what it means for something to be algorithmic. The need then did arise, in those years, in connection with attempts to prove that for some problems no algorithmic solution is possible. In 1935 Alan Turing considered this matter in isolation at Cambridge University. Meanwhile, in Princeton, the combined efforts of Kurt Gödel and Alonzo Church with his students Stephen Kleene and J. Barkley Rosser were brought to bear on the same topic. E.L. Post had also been thinking about these things, like Turing also in isolation, since the 1920s. Although the various formulations that were developed seemed superficially to be quite different from one another, they all turned out to be equivalent. This consensus about algorithmic processes has come to be called the Church–Turing Thesis.

Turing's conception differed from all the others in that it was formulated in terms of an abstract machine. What was striking about his characterization is his analysis that showed why even very limited fundamental operations would suffice for all algorithms if supplemented by unlimited data storage capability. Moreover, he demonstrated that a single such machine, Turing's so-called 'Universal Machine', could be made to carry out any algorithmic process whatever. These insights have played a key role in the development of modern all-purpose computers. But, as will become clear later, the notion of universality spills over into mathematical domains far removed from computation.[1]

Turing began his investigation with a problem from mathematical logic, a prob-

Published in *The Once and Future Turing*, edited by S. Barry Cooper & Andrew Hodges. Published by Cambridge University Press © 2016. Not for distribution without permission.

[1] The paper Turing (1936) has been reprinted in numerous collections.

lem, whose importance was emphasized by David Hilbert, that can be described as follows:

> Find an algorithm that will determine whether a specified conclusion follows from given premises using the formal rules of logical reasoning.

By 1935 there were good reasons to believe that no such algorithm exists, and this is what Turing set out to prove. Turing succeeded by first finding an unsolvable problem in terms of his machines, specifically the problem of determining for a given such machine whether it would ever output the digit 0. Then he showed how to translate this so-called "printing" problem into the language of mathematical logic in such a way that a purported algorithm for the problem from logic could be used to obtain a corresponding algorithm solving this printing problem, something he had shown could not be done. Later Post used the unsolvability of the printing problem to prove that a previously posed mathematical problem called the *word problem for semigroups* is likewise unsolvable. Still later, Turing himself used an intricate construction to refine and extend Post's result. Turing's beautiful article (Turing, 1954) explained unsolvability to the general public in clear simple terms.

When Turing learned that work leading to conclusions similar to his own had been done in Princeton, he arranged a visit and spent two years there. A new branch of mathematics variously called *computability theory, recursive function theory* or *recursion theory* was now open to vigorous pursuit. One direction this new discipline took was to lead to unsolvability proofs for problems from various branches of mathematics. This essay will tell the story of where one such thread led.

The Plot

Our discussion will be framed in terms of the so-called natural numbers $0, 1, 2, 3, \ldots$ We will work with three different ways to specify a set of natural numbers:

(1) by an algorithm that lists all the members of the set;
(2) by an algorithm to decide membership in the set;
(3) as the parameter values for which an equation has solutions.

It will turn out that examining how these notions are related to one another will lead to surprising and far reaching conclusions.

In 1949, in my graduate student days, I conjectured that two of these three are equivalent in the sense that the sets of natural numbers specified by them are the same. Although the truth of this conjecture could be seen to have very important consequences, it was generally regarded as quite implausible. I hardly imagined then that the proof of my conjecture would require two decades of work involving one of America's most prominent philosophers, the first female mathematician

to be elected to the National Academy of Science of the United States, a young Russian mathematician, and myself.

Although a mathematically rigorous treatment would require working with the technical notions developed by Turing and the others mentioned above, for the most part in this essay, algorithms will be dealt with in a loose informal manner,

An Example: The Set of Perfect Squares

A number obtained by multiplying some natural number by itself is called a *perfect square*. So the set of perfect squares is $\{0,1,4,9,16,25,\ldots\}$.

Algorithm for *listing* the perfect squares

```
Start with 0 and repeatedly add 1 to it generating the sequence of
all natural numbers. As each number is generated, multiply it by
itself and put the result in a list.
```

Note that there are infinitely many perfect squares and so the imagined computation to list them will continue 'forever'. The table below shows the list in the second row:

0	1	2	3	4	5	…
0	1	4	9	16	25	…

Definition A set of natural numbers is **listable** if there is an algorithm that lists its members (in any order with repetitions permitted).

Algorithm for *deciding* membership in the set of perfect squares

To decide whether a given number n is a perfect square, begin as above listing the perfect squares in order. If one of the perfect squares is equal to n, we can stop and we know that n is a perfect square; if one of the perfect squares is larger than n, we can stop and we know that n is not a perfect square.

Definition A set of natural numbers is **decidable** if there is an algorithm that decides membership in it.[2]

Using an equation to specify the set of perfect squares

In the equation

$$a - x^2 = 0$$

we regard a as a *parameter* and x as an *unknown*. This means that for different values of a we seek a natural number value of x that satisfies the equation. Since

[2] Other terms used for 'listable' are: recursively enumerable, computably enumerable. Other terms for 'decidable' are: computable, recursive.

this equation is obviously equivalent to $a = x^2$, the values of a for which such a solution exists are exactly the perfect squares.

Generally when we use the word "equation" we have in mind a polynomial expression set equal to 0. Polynomial expressions can involve any number of unknowns and may also include the parameter a. The expressions can be formed by combining the letters and any number of natural number constants using addition, subtraction, and multiplication. Here are some examples of polynomial expressions:

$$x^2 - 17y^2 - 2, \qquad x^3 - x^2y + 3axy^2, \qquad (16 + a^3)(ax + y^5).$$

Definition A set of natural numbers is **Diophantine** if there is a polynomial equation with parameter a that has natural number solutions for exactly those values of a that are members of the set.

Examples of Diophantine Sets

- $a - (x+2)(y+2) = 0$ specifies the set of *composite numbers*; that is, numbers other than 1 that are not primes.
- $a - (2x+3)(y+1) = 0$ specifies the set of numbers that are not powers of 2 (because they are the numbers other than 1 that have an odd divisor).
- The so-called *Pell equation*, $x^2 - a(y+1)^2 - 1 = 0$, has been well studied. It can be proved that in addition to the obvious solutions when $a = 0$ it has solutions precisely when a is **not** a perfect square.

Some Relationships

Theorem *Every decidable set is listable.*

Proof Let S be a decidable set. Generate the natural numbers in order. As each number is generated, test it to see whether it belongs to S. If it does, place it on a list. Move on to the next natural number. This algorithm will make a list of the members of S. □

The *complement* of a set S, written $\bar{S}$, is the set of all natural numbers that don't belong to S.

Theorem *The set S is decidable if and only if $S, \bar{S}$ are both listable.*

Proof If S is decidable then obviously so is $\bar{S}$, and hence both are listable.

On the other hand, if $S, \bar{S}$ are both listable then, given a number n, we can decide whether it belongs to S as follows. We use the two listing algorithms to start making

lists of both S and $\bar{S}$. We wait to see in which list n will eventually show up. Then we will know whether n belongs to S. □

Unsolvability Theorem *There is a listable set K whose complement $\bar{K}$ is not listable. Therefore K is not decidable.*

Proof See Appendix.

Theorem *Every Diophantine set is listable.*

Proof Let S be a Diophantine set, specified by an equation with parameter a and unknowns $x_1, \ldots, x_k$. Set up an ordering of all $(k+1)$-tuples $\langle a, x_1, \ldots, x_k \rangle$ of natural numbers,[3] and proceed through them one by one. For each tuple, plug the numbers into the equation. Checking to see whether these numbers do satisfy the equation is just a matter of arithmetic. If the equation is satisfied by a particular tuple, place the value of a from that tuple on a list. This algorithm will make a list of the members of S. □

A Conjecture Becomes a Theorem: a Story

When I wrote my doctoral dissertation in 1950, I stuck my neck out with following:

Conjecture *Every listable set is Diophantine.*

On the face of it this was quite implausible. Why should any set that can be listed by an algorithm be specifiable by something as simple as a polynomial equation? Moreover, for reasons that will be explained later, the conjecture implies something really implausible, that there are constants m and n such that every Diophantine set can be specified by an equation of degree $\leq m$ and with a number of unknowns $\leq n$. However, the conjecture, if true, can be seen to lead to a solution of one of the famous Hilbert problems.

At an International Congress of Mathematicians in 1900, David Hilbert listed 23 problems as a challenge for the future. These have become known as the Hilbert Problems and, in addition to their intrinsic interest, these problems have commanded special attention because of the stature of their source. The tenth problem in the list can be stated as follows:

[3] One way to order the tuples is to introduce each natural number in succession, writing down each tuple that includes that number together with all the previous numbers. For the case of one unknown, the pairs ordered in that manner look like this:

$$\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 0,2\rangle, \langle 2,0\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle, \ldots$$

Problem *Find an algorithm to determine for any given polynomial equation with integer coefficients whether it has a natural number solution.*[4]

It isn't difficult to see that the truth of my conjecture would solve Hilbert's tenth problem in a negative way, by implying that no such algorithm exists. The reason is that my conjecture implies that the set K, from the Unsolvability Theorem above, is Diophantine. So there would be an equation that would have solutions for a given value of its parameter a just in the case where a belongs to K. Thus, if there were an algorithm such as Hilbert had asked for, it could be used to decide membership in K, contradicting the fact that K is not decidable.

Despite its apparent implausibility, I had a reason to think that my conjecture might actually be true. I was able to prove that there is a Diophantine set S whose complement $\bar{S}$ is not Diophantine. The comparison with the Unsolvability Theorem is striking. My proof was short and easy, but not *constructive*. (That means that the proof did not provide any example of such a set; it merely proved its existence.) It was easy to see that Diophantine sets as a class shared other properties with the listable sets.[5]

I tried to prove my conjecture, but the best I could do, reported in my dissertation, was to prove that for every listable set S, there is an equation with parameters a, q and k such that a number a belongs to S if and only if there is a number q_0 such that the equation has solutions for that value of a, for $q = q_0$, and for all values of $k \leq q_0$.[6] Although far from what I desired, this result turned out to play a significant role in what followed.

When I attended the International Congress of Mathematicians at Harvard University in 1950, I learned that Julia Robinson, a mathematician with whose work I was familiar, had also been working on Diophantine sets. But whereas I had been working top down, trying to get a Diophantine-like representation for listable sets, she had been working bottom up, trying for Diophantine definitions of various sets. Alfred Tarski had suggested that one should prove that the set of powers of 2, $\{1, 2, 4, 8, 16, \ldots\}$, is *not* Diophantine. Julia was attracted to this problem but, not succeeding in doing what Tarski had proposed, turned around and tried to prove that the set of powers of 2 *is* Diophantine. Of course, had Tarski been right it would have shown that my conjecture is false. Julia couldn't prove that this set is Diophantine either. But she did prove that if one could find what I like to call a Goldilocks

[4] Actually, Hilbert asked for an algorithm for arbitrary integer solutions, positive, negative or zero. However, it is not difficult to see that the two forms of the problem are equivalent.

[5] For example, the union as well as the intersection of two Diophantine sets is also Diophantine.

[6] Using logical symbolism,

$$a \in S \Leftrightarrow (\exists q)(\forall k)_{\leq q}(\exists x_1, \ldots x_n)[p(a, k, q, x_1, \ldots x_n) = 0],$$

where p is a polynomial with integer coefficients.

equation then she could prove not only that the set of powers of 2 is Diophantine but that the set of prime numbers, as well as many other sets, are Diophantine.

In the folktale, Goldilocks experiences the bears' accommodations and finds those that are too large, those that are too small, and finally those that are 'just right'. Here we are considering equations with two parameters, a and b. Such an equation is *too large* if there are values of a, b with $b > a^a$ for which the equation has solutions. It is *too small* if there is a number k, such that for all values of a, b for which the equation has solutions, $b \leq a^k$. So an equation is a *Goldilocks equation* if it is *just right*, neither too large nor too small. Julia tried to find such a Goldilocks equation, but she did not succeed.

It was at a month-long 'Institute for Logic' at Cornell University in the summer of 1957 that Hilary Putnam and I began working together. We did get some preliminary results that summer, but our breakthrough occurred two years later. Our idea was to see what would happen to my conjecture if we permitted variable exponents in the equation. Thus in addition to equations like $x^5y - 7y^3a^2z + 5 = 0$ we would also be considering equations like $x^5y^z - 7y^3a^2z + 5 = 0$. Although we did begin with my dissertation work, the introduction of variable exponents brought us squarely into Julia Robinson's territory, and we found ourselves using generalizations of some of her methods. Sets specified by equations with a parameter, where variable exponents are allowed, are called *exponential Diophantine*. We were trying to prove that

Every listable set is exponential Diophantine. $(***)$

We came close. But we had to make use of a property of prime numbers that was believed to be true but had not yet been proved:

PAP *For every number n, there is a prime number p and a number k such that the numbers $p, p+k, p+2k, \ldots, p+nk$ are all prime.*

What we were able to prove was only that:

Theorem *If PAP is true then $(***)$ is also true.*

Actually we now know that PAP is true, because it was proved in 2004, so in a sense our proof of (***) was correct. But in 1959, having no access to a time machine, we had to content ourselves with the mere implication.

The existence of a Goldilocks equation also came into the picture. It follows easily from Julia's work that if there is a Goldilocks equation then every exponential Diophantine set is also Diophantine. Hilary and I introduced the abbreviation:

JR *There exists a Goldilocks equation.*