Fundamenta Informaticae 56 (2003) 273–284
IOS Press

On the existence of a new family of Diophantine equations for Ω

Toby Ord
Department of Philosophy
The University of Melbourne
Parkville 3010, Australia
t.ord@pgrad.unimelb.edu.au

Tien D. Kieu[*]
Centre for Atom Optics and Ultrafast Spectroscopy
Swinburne University of Technology
Hawthorn 3122, Australia
kieu@swin.edu.au

Abstract. We show how to determine the k-th bit of Chaitin's algorithmically random real number Ω by solving k instances of the halting problem. From this we then reduce the problem of determining the k-th bit of Ω to determining whether a certain Diophantine equation with two parameters, k and N, has solutions for an odd or an even number of values of N. We also demonstrate two further examples of Ω in number theory: an exponential Diophantine equation with a parameter k which has an odd number of solutions iff the k-th bit of Ω is 1, and a polynomial of positive integer variables and a parameter k that takes on an odd number of positive values iff the k-th bit of Ω is 1.

Keywords: Diophantine Equation, Ω, Algorithmic Information Theory, Randomness, Hilbert's Tenth Problem

1. Introduction and Motivation

In his 1975 paper [3], Gregory Chaitin introduced the number Ω. Ω is a real number between 0 and 1 which is the *halting probability*: the probability that a randomly chosen program will halt. In order to

[*]Address for correspondence: Centre for Atom Optics and Ultrafast Spectroscopy, Swinburne University of Technology, Hawthorn 3122, Australia

make this precise, one must choose a particular (recursive) programming language in which the programs are self-delimiting binary strings. This allows us to choose a random program by flipping an unbiased coin to generate each bit, stopping when we reach a valid (self-delimited) program. This method makes longer programs increasingly unlikely and allows Ω to be a well defined probability. The exact value of Ω obviously depends upon which programming language we choose, and thus Chaitin [4] specifies a particular language to make talk of Ω concrete. However, its important properties are the same regardless and the exact language chosen need not concern us.[1]

It is well known that if we had access to Ω we could solve the halting problem [4]. Furthermore, if we had some method of solving the halting problem then we could compute Ω to any desired accuracy. In this sense there is an equivalence between Ω and the halting problem: the ability to compute one gives us the ability to compute the other.

It is also well known that both the halting problem and Ω have analogues in number theory. The halting problem has been shown to be equivalent to the problem of determining whether a given Diophantine equation has solutions, while Ω has been shown to be equivalent to determining whether a finite or infinite amount of Diophantine equations of a particular type have solutions. However, unlike their counterparts in the theory of computability, these number theoretic versions have no direct connection between them. In this paper, we present a new representation of Ω within number theory that revolves around a matter of parity rather than finitude and makes clear the link between the number theoretic analogues of Ω and the halting problem.

In Sections 2 and 3 we provide the formal definitions of Ω and a related real number τ, presenting an efficient method for using the solutions to the halting problem to determine Ω. In Sections 4 and 5, we then introduce Diophantine equations and review the known method of using them to represent Ω, providing proofs when they will facilitate the proofs of our new results. In Sections 6 and 7 we present and prove our results for a new representation of Ω and discuss their implications.

2. Algorithmic Randomness and Ω

Definition 2.1. Ω is formally defined by the following equation, where p ranges over all programs in the language specified by Chaitin [4] and $|p|$ is the number of digits in the binary representation of p.

$$\Omega = \sum_{p \text{ halts}} 2^{-|p|} \tag{1}$$

The importance of Ω lies in its contrasting properties of being both an *algorithmically random* and *recursively enumerable* real number.

As a random real, Ω is uncomputable in a very strong sense. If we look at the binary expansion of Ω (the infinite sequence of 1's and 0's following the binary point), this sequence is highly incompressible. To see how this is so, let us compare it with the number τ from Copeland and Proudfoot [6], a non-recursive real that is not random.

Definition 2.2. τ is the real number between 0 and 1 whose n-th binary digit is 1 if the n-th program p_n halts when given no input and 0 otherwise. In order to pick out a particular ordering of programs, let

[1]The properties possessed by Ω are numerous and we can only discuss the most pertinent here. For further information on Ω's exotic attributes, see [4, 10, 1, 2].

each program have a binary representation as above and let them be ordered in the usual lexical ordering for binary strings. We could also write:

$$\tau = \sum_{p_n \text{ halts}} 2^{-n} \tag{2}$$

τ in an encoding of the answers to every instance of the halting problem in a single real number. While each bit of τ tells us whether a particular program halts when run with no input, we can easily use the bits of τ to answer the more common formulation of halting problem: 'does program p halt when run on input i?'. Given p and i, we can (recursively) construct a new program which takes no input and simulates the computation of p on i. The bit of τ corresponding to this new program tells us whether or not p halts on i.

The concept of compressibility for an infinite string can be understood through examining the minimum amount of advice needed to find out n bits of that string [4]. For recursive infinite strings (such as the binary digits of π), we can perform massive compression by asking for the bits of a particular program which will generate every bit of π, one by one. In this sense, there are only finitely many bits of information (those of that particular program) in the infinite bitstring that makes up π.

On the other hand, if we wish to determine the bits of τ, there is no program that will do this, since the halting problem is undecidable. Therefore, determining n bits of τ will require an amount of advice that increases with the value of n. Clearly we could find out n bits of τ with n bits of advice as we could be directly given those bits. However, Chaitin [4] shows how we can do better than this, using only $\log_2 n$ bits. This is because the n bits of τ represent n instances of the halting problem and to solve these, we only need to know how many of the n programs halt. We can then simulate each program in parallel until this many have halted, being confident that every one that is still running will never halt. Therefore, while the (Turing) non-computability of τ makes it globally incompressible (computing infinitely many bits of τ requires infinitely many bits of advice), it is still locally compressible as n bits of τ are computable from $\log_2 n$ bits of advice.

In contrast, Chaitin [4] defines a random real as one for which calculating n bits of its binary expansion requires more than n bits of advice. The reason that *more* than n bits are needed is that the advice, just like the program itself, must be self-delimiting. Chaitin [3] has shown that Ω satisfies this condition and is thus a random real. In this way, the first n bits of Ω contain n bits of *algorithmically incompressible* information.

In addition to recursive incompressibility, random reals are also characterised by *recursive unpredictability* [4]. Consider a 'predictive' program that takes a finite initial segment of an infinite bitstring and returns a value indicating either 'the next bit is 1', 'the next bit is 0' or 'no prediction'. If any such program is run on all finite prefixes of the binary expansion of a random real and makes an infinite amount of predictions, the limiting relative frequency of correct predictions approaches $\frac{1}{2}$. In other words when any program is used to predict infinitely many bits of a random real, such as Ω, it does no better than random — even with information about all the prior bits.

In contrast to these results, Ω is a recursively enumerable (r.e.) real. For a mathematical object to be r.e. it need not be computable, but there must be a certain method of successively approximating it. For a set of positive integers to be r.e. there must be a program that halts on i if and only if i is in the set. For an infinite binary sequence to be r.e. there must be a program that halts on i if and only if the i-th bit of the sequence is 1. For a real number x to be r.e. there must be a program that takes a positive integer i and generates a rational approximation x_i where $\{x_i\}$ is an increasing sequence of rationals that converges to x. In the case of Ω, we can construct such a sequence which we shall denote $\{\Omega_i\}$, where

$$\Omega_i = \sum_{\substack{|p| \leq i \\ p \text{ halts in } \leq i \text{ steps}}} 2^{-|p|} \tag{3}$$

However, it is also important to ask whether the binary expansion of Ω is an r.e. bitstring. For τ, this is the case because there is a program that halts on i if and only if the i-th bit is 1 — indeed this is just a universal program, which simulates the i-th program (and thus halts iff the i-th program does). For Ω however, there is no such program. This is a rather surprising, but crucial, fact about Ω.

We can see that this must be true given the incompressibility of Ω. If there were a program to enumerate the bits of Ω, we could use the trick described above for compressing τ: we could determine n bits of Ω by asking how many of these n values make this program halt (and thereby require only $\log_2 n$ bits). Therefore, as Ω is incompressible, we see that there is indeed no such program for Ω and that its sequence of bits is not r.e. Because all random reals share the property of being incompressible, this argument carries over and no random real can have an r.e. sequence of bits.

In the remainder of the paper, we shall use the fact that Ω is an r.e. real to express it through Diophantine equations and thereby show how algorithmic randomness occurs even in the heart of number theory.

3. Computing Ω from τ

Having seen that there are interesting connections between Ω and τ, it is natural to ask whether we can compute Ω given access to τ (or in the language of recursion theory, whether Ω is *Turing reducible* to τ). One way to go about this is to construct a program Q that takes two positive integers, k and N, as input and halts iff $\frac{N}{2^k} < \Omega$. Such a program is quite easy to construct, it just needs to enumerate $\{\Omega_i\}$ (as done in the previous section) and check at each stage whether $\frac{N}{2^k} < \Omega_i$, halting if this is true and continuing otherwise. Since $\{\Omega_i\}$ approaches Ω from below, if $\frac{N}{2^k} < \Omega$ then there is some i for which $\frac{N}{2^k} < \Omega_i$ and Q halts as required. On the other hand, if $\frac{N}{2^k} \geq \Omega$ then there is no such i and Q will not halt. From τ, we can determine whether or not Q halts on a given input, and thus whether or not each $\frac{N}{2^k}$ is less than Ω.

To determine the first k bits of Ω, we just need to determine the greatest value of N for which $\frac{N}{2^k} < \Omega$. We will then know that for this N, $\frac{N}{2^k} < \Omega < \frac{N+1}{2^k}$ and thus that if N is expressed as a k digit binary number (with leading zeros if required), it will be the first k digits of Ω.

We can therefore use τ to determine the first k bits of Ω, by checking whether Q halts when applied to k and N for each value of N from 1 to $2^k - 1$. This technique allows us to determine the first k bits of Ω from $2^k - 1$ carefully chosen bits of τ. Indeed, given these bits, it is so simple to compute the first k bits of Ω, that we could do it by simply combining them with a truth table and Ω is not only Turing reducible to τ, but also *truth table reducible*.

However, if we are prepared to sacrifice this extreme algorithmic simplicity, we can get by with even fewer bits of τ by using a *bisection search*. In the bits of τ that we need we know there is a lot of structure: if $\frac{N}{2^k} < \Omega$ then for all $i < N$, $\frac{i}{2^k} < \Omega$. Similarly, if $\frac{N}{2^k} \geq \Omega$ then for all $i > N$, $\frac{i}{2^k} \geq \Omega$. Thus, all we need to find is the greatest value of N for which $\frac{N}{2^k} < \Omega$. This can be done by first trying $N = \frac{2^k}{2}$

and checking if it is less than or greater than Ω. If it is greater, we then try the midpoint of 0 and $\frac{2^k}{2}$. If it is less, we try the midpoint of $\frac{2^k}{2}$ and 2^k. We proceed in this manner until finding the value of N for which $\frac{N}{2^k} < \Omega$ and $\frac{N+1}{2^k} \geq \Omega$.

Performing this type of bisection search on n items requires only $\log_2(n+1)$ queries, rounding up to the nearest integer. As we do not need to test for $N = 0$ or $N = 2^k$, there are only $2^k - 1$ remaining values to check and we can therefore get the first k bits of Ω with only k bits of τ. From the incompressibility of Ω, we can see that this must be optimal: no less than k bits of any other real whatsoever can give us k bits of Ω.[2]

Having shown a method for determining the bits of Ω from an r.e. sequence of bits (τ), we will now examine Diophantine equations and look at how this method can be used to find Ω in number theory.

4. Diophantine Equations and Hilbert's Tenth Problem

Definition 4.1. A *Diophantine equation* is an equation of the form

$$D(x_1, \ldots, x_m) = 0 \tag{4}$$

where D is a polynomial with integer coefficients. The variables are typically allowed to range over the integers or the non-negative integers, but here it will be convenient to restrict them to the positive integers.

The study of Diophantine equations is a central area of number theory. It is well known that some Diophantine equations have solutions (such as $2x_1 - 4 = 0$) and some do not (such as $2x_1 - 3 = 0$). The task of devising an algorithm for determining whether or not an arbitrary Diophantine equation is solvable is known as Hilbert's tenth problem and has been a major area of research in 20th Century mathematics (see Matiyasevich [8]).

Definition 4.2. A *family of Diophantine equations* is a relation of the form

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0 \tag{5}$$

where we distinguish between the variables $a_1, \ldots, a_n$ which are called parameters and $x_1, \ldots, x_m$ which are called unknowns. Fixing values of the parameters, specifies one of the individual Diophantine equations that comprise the family.

Definition 4.3. A set of n-tuples, $\mathcal{D}$, is *Diophantine* iff it can be represented by a family of Diophantine equations in the following way

$$\langle a_1, \ldots, a_n \rangle \in \mathcal{D} \iff \exists x_1 \ldots x_m D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0 \tag{6}$$

[2]It may seem like there is a contradiction here because we could use the compression trick for τ again, giving us a method for computing k bits of Ω from only $\log_2 k$ bits of advice. However, this turns out to be impossible because to use the bisection search, we need to find out the value of one bit of τ before we know which bit to ask for next. There is no time at which we know in advance a complete set of k bits of τ that we can ask for in compressed form.

Many sets, such as the square numbers or the pairs $\langle a, b\rangle$ such that a divides b are Diophantine. The question of exactly which sets are Diophantine has received considerable attention and was solved through the work of Martin Davis, Hilary Putnam, Julia Robinson and Yuri Matiyasevich (DPRM) [8].

Theorem 4.1. (DPRM Theorem)
Every r.e. set $\mathcal{D}$ of n-tuples of positive integers is Diophantine, that is:

$$\langle a_1, \ldots, a_n\rangle \in \mathcal{D} \iff \exists x_1 \ldots x_m D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0 \tag{7}$$

Proof:
See [8]. □

The DPRM Theorem provides an excellent tool for showing the existence of Diophantine equations with certain properties. All that is needed is to come up with a set that has the desired property and prove that it is r.e. For example, the following is immediately evident:

Corollary 4.1. (Family of Diophantine equations for τ)
There is a family of Diophantine equations:

$$T(n, x_1, \ldots, x_m) = 0 \tag{8}$$

which has solutions for a given value of n iff the n-th bit of τ is 1 or, equivalently, iff the n-th program halts.

Proof:
The set of positive integers n such that the n-th bit of τ is 1 is clearly an r.e. set. Thus, by the DPRM Theorem, there is a family of Diophantine equations with a parameter, n, which has a solution for a given value of n iff the n-th bit of τ is 1. □

From this it is clear that Hilbert's tenth problem must be recursively undecidable. No program could decide whether a given Diophantine equation has a solution because this would allow a program to compute the bits of τ — a task that is known to be non-recursive.[3] We therefore have undecidability in number theory, with direct analogues of the halting problem and τ. In what follows we show how we can also find algorithmic randomness and, in doing so, add Ω to this list.

5. Chaitin's Expression of Ω Through Diophantine Equations

From the DPRM Theorem, we can see that Ω cannot be directly represented by a Diophantine set (one that has k as a member iff the k-th bit of Ω is 1) because the bits of Ω are not r.e. However, taking a slightly different approach, Chaitin [4] shows that the bits of Ω can be found in another property of Diophantine equations.

[3]Indeed, it was long known that the recursive undecidability of Hilbert's Tenth Problem would follow immediately from the DPRM Theorem and this was the main motivation for its proof — the Diophantine representations for all other r.e. sets being a nice corollary.

Theorem 5.1. (Family of Diophantine Equations for Ω via Finitude)
There is a family of Diophantine equations

$$\chi_1(k, N, x_1, \ldots, x_m) = 0 \tag{9}$$

such that for a given k there are infinitely many values of N for which it has a solution if the k-th bit of Ω is 1 and finitely many values of N for which it has a solution if the k-th bit of Ω is 0.

Proof:
There is a program P that takes two inputs, k and N, generating Ω_N and returning the value of its k-th bit. As N increases, the k-th bit of Ω_N may change between 0 and 1 many times, but since $\{\Omega_N\}$ is monotonically increasing, each bit can only change finitely many times (at most $2^k - 1$). Therefore, the k-th bit of Ω_N will eventually settle on either a 0 or a 1 and because $\{\Omega_N\}$ converges to Ω, this final value must be the same as the k-th bit of Ω. Thus, the k-th bit of Ω_N will only differ from the k-th bit of Ω for finitely many values of N.

Let $\mathcal{C}_1$ be the set of pairs $\langle k, N\rangle$ such that P returns 1 when applied to k and N. If the k-th bit of Ω is 0, P will return 1 (the correct value for Ω_N but incorrect for Ω) for only finitely many values of N. On the other hand, if the k-th bit of Ω is 1, P will return 1 for infinitely many values of N. In other words, for a given k, there are infinitely many pairs $\langle k, N\rangle \in \mathcal{C}_1$ if the k-th bit of Ω is 1 and finitely many if it is 0. Since $\mathcal{C}_1$ is recursive, it is r.e. and thus via the DPRM Theorem there is a Diophantine equation, $\chi_1(k, N, x_1, \ldots, x_m) = 0$, that has a solution iff $\langle k, N\rangle \in \mathcal{C}_1$. Thus, for a given value of k, this equation has solutions for infinitely many values of N iff the k-th bit of Ω is 1. □

Chaitin takes this result further and provides a simpler example of Ω occurring in number theory. In doing so he moves to *exponential Diophantine equations*. Where the polynomial in a Diophantine equation consists of variables and integer constants composed together with addition and multiplication, exponential Diophantine equations merely add exponentiation to this list, with the proviso that no negative constants can appear in the exponents. In proving the DPRM Theorem, Matiyasevich [8] showed that all sets which are exponential Diophantine are Diophantine as well. However, Chaitin's move to exponential Diophantine equations allows the use of another key result due to Matiyasevich [8] which concerns not just the existence of solutions to an equation, but the quantity of solutions as well.

Theorem 5.2. (Existence of Singlefold Exponential Diophantine Equations)
For any Diophantine set, $\mathcal{D}$, there is a family of exponential Diophantine equations

$$E(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0 \tag{10}$$

that is a *singlefold* representation of $\mathcal{D}$: it has exactly one solution for $\langle a_1, \ldots, a_n\rangle \in \mathcal{D}$ and no solutions otherwise.

Proof:
See [8]. □

Theorem 5.3. (Family of Exponential Diophantine Equations for Ω via Finitude)
There is a family of exponential Diophantine equations

$$\chi_1^e(k, x_0, x_1, \ldots, x_m) = 0 \tag{11}$$

with infinitely many solutions for k if the k-th bit of Ω is 1 and finitely many if the k-th bit of Ω is 0.

Proof:
Let $\chi_1^e(k, N, x_1, \ldots, x_m) = 0$ be the family of singlefold exponential Diophantine equations for χ_1 of Theorem 5.1. The existence of χ_1^e is ensured by Theorem 5.2. Given a value of k, there is a single solution for each of infinitely many values of N iff the k-th bit of Ω is 1. If we treat N as another unknown instead of a parameter, we get $\chi_1^e(k, x_0, x_1, \ldots, x_m) = 0$ which is a family of exponential Diophantine equations with infinitely many solutions for k iff the k-th bit of Ω is 1. □

Thus, while the question of whether a Diophantine equation has solutions is undecidable in general, the question of whether an exponential Diophantine equation has infinitely many solutions is much worse. The task is no longer r.e. (since solving it gives the bits of Ω) and as a single parameter is varied, the results fluctuate in an algorithmically random manner: there is absolutely no recursive pattern to be found.

Chaitin [4] went even further than we have here by actually constructing an exponential Diophantine equation $\chi_1^e(k, x_0, x_1, \ldots, x_m) = 0$. This equation was automatically generated from a complex register machine program and is very large, with approximately 17,000 unknowns. While it is has been shown that this can be reduced to just three [8], doing so would be a very challenging task.

6. A New Expression of Ω Through Diophantine Equations

Using the method of computing Ω from τ discussed in Section 3 we can now present our main results. While the bits of Ω occur in Theorems 5.1 and 5.3 through the distinction between the finite and the infinite, the presentation below remains within the realms of the finite, with the bits of Ω occurring in the changes of parity.

Theorem 6.1. (Family of Diophantine Equations for Ω via Parity)
There is a family of Diophantine equations

$$\chi_2(k, N, x_1, \ldots, x_m) = 0 \tag{12}$$

such that for a given k:

- there are less than 2^k values of N, taken from $\{1, \ldots, 2^k - 1\}$, for which there is a solution,
- and there are an odd number of values of N for which there is a solution iff the k-th bit of Ω is 1.

Proof:
Let $\mathcal{C}_2$ be the set of pairs of positive integers $\langle k, N\rangle$ such that $\frac{N}{2^k} < \Omega$. From Section 3, we see that $\mathcal{C}_2$ is r.e. and so, by the DPRM Theorem, there exists a Diophantine equation, $\chi_2(k, N, x_1, \ldots, x_m) = 0$, which has a solution iff $\langle k, N\rangle \in \mathcal{C}_2$. Since $0 < \Omega < 1$, $\langle k, N\rangle \in \mathcal{C}_2$ implies that $N \in \{1, \ldots, 2^k - 1\}$, and so these are the only values of N for which $\chi_2(k, N, x_1, \ldots, x_m) = 0$ can have solutions.

If the greatest value of N for which $\frac{N}{2^k} < \Omega$ is odd, then the k-th bit of Ω must be 1 and similarly, if the greatest value of N for which $\frac{N}{2^k} < \Omega$ is even, then the k-th bit of Ω must be 0. As the greatest value of N for which $\frac{N}{2^k} < \Omega$ is equal to the number of values of N for which $\frac{N}{2^k} < \Omega$, it follows that for a given k, there are an odd number of values of N for which $\chi_2(k, N, x_1, \ldots, x_m) = 0$ has solutions iff the k-th bit of Ω is 1. □

Unlike the representation of Ω through finitude, this representation would allow Ω to be directly computed by determining whether certain Diophantine equations have solutions (i.e. by solving several instances of Hilbert's tenth problem). This mirrors the relationship between Ω and the halting problem in computability theory. Even the efficiency of the computation is preserved: the bisection approach could also be used here to determine the k-th bit of Ω from the solutions to just k instances of Hilbert's tenth problem.

Theorem 6.2. (Family of Exponential Diophantine Equations for Ω via Parity)
There is a family of exponential Diophantine equations

$$\chi_2^e(k, x_0, x_1, \ldots, x_m) = 0 \tag{13}$$

which for a given k,

- has less than 2^k solutions, where x_0 takes distinct values from the set $\{1, \ldots, 2^k - 1\}$,
- and has an odd number of solutions iff the k-th bit of Ω is 1.

Proof:
Let $\chi_2^e(k, N, x_1, \ldots, x_m) = 0$ be the singlefold family of exponential Diophantine equations for $\mathcal{C}_2$. Given a value of k, there is a single solution for each of an odd number of values of N iff the k-th bit of Ω is 1. If we treat N as another unknown instead of as a parameter, we get $\chi_2^e(k, x_0, x_1, \ldots, x_m) = 0$. For a given value of k, this has less than 2^k solutions and in these solutions, x_0 takes distinct values from $\{1, \ldots, 2^k - 1\}$. Furthermore, the number of solutions is odd iff the k-th bit of Ω is 1. $\square$

In addition to the above results, we can use both the finitude and parity methods to produce a new source of algorithmic randomness in number theory. Using the following lemma [8], we can design polynomial expressions with variables and a parameter k ranging over the positive integers that exhibit randomness in the number of positive values they assume as k is varied.

Lemma 6.1. Given a family of Diophantine equations with two parameters

$$D(k, N, x_1, \ldots, x_m) = 0 \tag{14}$$

we can construct a polynomial with integer coefficients, one parameter and one additional variable x_0 as follows

$$P(k, x_0, x_1, \ldots, x_m) \equiv x_0 \left(1 - (D(k, x_0, x_1, \ldots, x_m))^2\right). \tag{15}$$

If we restrict the variables to values in the positive integers, then the set of positive values that this polynomial assumes for a given k is exactly the set of all N such that $D(k, N, x_1, \ldots, x_m) = 0$ has solutions.

Proof:
For all $x_0, x_1, \ldots, x_m$ such that $D(k, x_0, x_1, \ldots, x_m) = 0$, the polynomial $P(k, x_0, x_1, \ldots, x_m)$ will assume the value x_0. For all $x_0, x_1, \ldots, x_m$ such that $D(k, x_0, x_1, \ldots, x_m) \neq 0$, $P(k, x_0, x_1, \ldots, x_m)$ will assume a non-positive value. Thus, $P(k, x_0, x_1, \ldots, x_m)$ will assume all values of N such that $D(k, N, x_1, \ldots, x_m) = 0$ has solutions and no other positive values. $\square$

Theorem 6.3. (Polynomial Expression for Ω via Finitude)

$$P_1(k, x_0, x_1, \ldots, x_m) \equiv x_0 \left(1 - (\chi_1(k, x_0, x_1, \ldots, x_m))^2\right) \tag{16}$$

is a polynomial with integer coefficients and a parameter k. If we restrict the variables to values in the positive integers, then this polynomial assumes infinitely many positive values iff the k-th bit of Ω is 1.

Proof:
By Lemma 6.1, the set of positive values that P_1 assumes for a given k is precisely the set of N such that $\chi_1(k, N, x_1, \ldots, x_m) = 0$ has solutions. By Theorem 5.1, this set has infinitely many members iff the k-th bit of Ω is 1, so P_1 must assume a countable infinity of positive values iff the k-th bit of Ω is 1. □

Theorem 6.4. (Polynomial Expression for Ω via Parity)

$$P_2(k, x_0, x_1, \ldots, x_m) \equiv x_0 \left(1 - (\chi_2(k, x_0, x_1, \ldots, x_m))^2\right) \tag{17}$$

is a polynomial with integer coefficients and a parameter k. If we restrict the variables to values in the positive integers, then

- this polynomial assumes less than 2^k values in the positive integers, taken from $\{1, \ldots, 2^k - 1\}$,
- and the number of positive values that it takes is odd iff the k-th bit of Ω is 1.

Proof:
From Lemma 6.1, the set of positive integer values that $P_2(k, x_0, x_1, \ldots, x_m)$ assumes for a given k is precisely the set of N such that $\langle k, N\rangle \in \mathcal{C}_2$. By Theorem 6.1, this set only includes values from $\{1, \ldots, 2^k - 1\}$. Theorem 6.1 also tells us that there are an odd number of N such that $\langle k, N\rangle \in \mathcal{C}_2$ iff the k-th bit of Ω is 1, so $P_2(k, x_0, x_1, \ldots, x_m)$ must assume an odd number of positive values iff the k-th bit of Ω is 1. □

While Theorems 6.1, 6.2 and 6.4 are presented in terms of producing the k-th bit of Ω, the ideas behind them have a somewhat more general application. In each theorem, a set of values is discussed and its size is considered. For Theorem 6.1, this is the number of values of N for which $\chi_2 = 0$ has solutions, for Theorem 6.2 it is the number of solutions for $\chi_2^e = 0$ and for Theorem 6.4 it is the number of positive values taken by P_2. For a given value of k, each of these quantities will equal the same number which we shall call q_k.

Previously, we just looked at the parity of q_k and used this single bit of information to determine a single bit of Ω. Since q_k may take on 2^k different values, it contains $k - 1$ additional bits of information and these can directly provide all prior bits of Ω. When q_k is expressed in binary (with enough leading zeros to give k digits) it comprises the first k bits of Ω. For example, if for $k = 5$, there are six solutions to $\chi_2^e = 0$, then the first five bits of Ω are 00110.

In this way, we can see that the first $k - 1$ bits of q_k are just the $k - 1$ bits of q_{k-1}. So, given q_{k-1}, there are only two possibilities for q_k: either $q_k = 2q_{k-1}$ or $q_k = 2q_{k-1} + 1$. Thus, while the sequence $\{q_k \mod 2\}$ is algorithmically random, the sequence $\{q_k\}$ is highly structured.

We can also extend our results to the expression of Ω in different bases. If we consider the representation of Ω in base b, we can generate an equation for finding its k-th digit by replacing all references to 2^k with b^k. Instead of checking the parity of the appropriate quantity, the k-th digit is simply the remainder when this quantity is divided by b. Similarly, if we represent the quantity in base b with enough leading zeros, it forms the first k digits in a base b representation of Ω and q_k must be one of $\{bq_{k-1}, bq_{k-1}+1, \ldots, bq_{k-1}+(b-1)\}$.

7. Concluding Remarks

In this paper we report on our findings on the existence of a new family of Diophantine equations for determining the digits of Ω. Our equations differ from the previous families of Diophantine equations in several important features. Previously, a particular bit of Ω was represented by whether some Diophantine equations have solutions for a finite or infinite number of values of a given parameter or, in the exponential Diophantine case, whether the number of solutions is finite. In contrast, our formulation always has a finite number of parameter values for which solutions could occur and a finite number of solutions in the case of the exponential Diophantine equations. The value of the corresponding bit of Ω depends only on whether this number is even or odd. Thus, in the relatively mundane switching between an odd or an even number of solutions as a parameter is varied, the full subtlety of algorithmic randomness is felt: the incompressibility and the unpredictability.

These new families of Diophantine equations for Ω also provide a symmetry to the relationship between randomness and undecidability in the fields of computability and number theory. Just as we can directly compute Ω (in its guise as the halting probability) from solutions to instances of the halting problem, so can we compute Ω (in its guise as a property of a family of Diophantine equations) from solutions to instances of Hilbert's tenth problem. Furthermore, the translation into the domain of number theory preserves the efficiency of the computation, producing n bits of Ω from either the solutions to n instances of the halting problem or the solutions to n instances of Hilbert's tenth problem.

Finally, we provide a further example of how algorithmic randomness occurs in number theory: a polynomial with positive integer variables and a parameter k that takes on an odd number of positive values iff the k-th bit of Ω is 1. Along with the other results of this paper, this hints at the variety of ways in which this randomness can occur. While the polynomials and exponentials that give rise to algorithmic randomness are not of the type that are likely to occur in classical research in number theory, their importance lies in showing that in some places there are facts with no recursive pattern at all. This is not to say that these facts are *completely* patternless — on the contrary, we have shown intricate dependencies between the expressions of Ω and those of the halting problem — but in the face of so subtle a pattern, computer programs can perform no better than coin tossing. Without some means to transcend the fundamental limits of our current computers, whether through some kind of mathematical insight or radical new technology[4], they will remain completely beyond our grasp.

Acknowledgements

We would like to thank Gregory Chaitin for helpful discussion. TDK wishes to acknowledge the continuing support of Peter Hannaford.

[4]Examples and discussions of such non-classical computation have been presented by both authors [7, 9] and a very thorough survey can be found in Copeland [5].

References

[1] Calude, C. S.: A Characterization of c.e. Random Reals, *Theoretical Computer Science*, **271**, 2002, 3–14.

[2] Calude, C. S., Nies, A.: Chaitin Ω Numbers and Strong Reducibilities, *Journal of Universal Computer Science*, **3**, 1997, 1162–1166.

[3] Chaitin, G. J.: A Theory of Program Size Formally Identical to Information Theory, *Journal of the ACM*, **22**(3), July 1975, 329–340.

[4] Chaitin, G. J.: *Algorithmic Information Theory*, Cambridge University Press, Cambridge, 1987.

[5] Copeland, B. J.: Hypercomputation, *Minds and Machines*, **12**, 2002, 461–502.

[6] Copeland, B. J., Proudfoot, D.: Alan Turing's Forgotten Ideas in Computer Science, *Scientific American*, April 1999, 77–81.

[7] Kieu, T. D.: Computing the Noncomputable, *Contemporary Physics*, **44**, 2003, 51–71.

[8] Matiyasevich, Y. V.: *Hilbert's Tenth Problem*, MIT Press, Cambridge, Massachusetts, 1993.

[9] Ord, T.: *Hypercomputation: Computing More Than the Turing Machine*, Technical report, University of Melbourne, Melbourne, Australia, September 2002, Available at http://www.arxiv.org/abs/math.LO/0209332.

[10] Solovay, R. M.: A Version of Ω for which ZFC Can Not Predict a Single Bit, in: *Finite Versus Infinite. Contributions to an Eternal Dilemma* (C. S. Calude, G. Păun, Eds.), Springer, London, 2000, 323–334.