

INFORMATION- THEORETIC INCOMPLETENESS

Applied Mathematics and Computation
52 (1992), pp. 83–101

G. J. Chaitin

Abstract

We propose an improved definition of the complexity of a formal axiomatic system: this is now taken to be the minimum size of a self-delimiting program for enumerating the set of theorems of the formal system. Using this new definition, we show (a) that no formal system of complexity n can exhibit a specific object with complexity greater than $n + c$, and (b) that a formal system of complexity n can determine at most $n + c$ scattered bits of the halting probability Ω . We also present a short, self-contained proof of (b).

Copyright © 1992, Elsevier Science Publishing Co., Inc., reprinted by permission.

1. Introduction

The main incompleteness theorems of my *Algorithmic Information Theory* monograph [16] are reformulated and proved here using a new and improved definition of the complexity of a formal axiomatic system. This new approach is the self-delimiting version of that used in my long 1974 paper [4], which may be contrasted with my short 1974 paper [3]. In other words, this paper and my monograph [16] stand in the same relation as my papers [4] and [3] do.

The new idea is to measure the complexity of a formal system in terms of the program-size complexity of enumerating its infinite set of theorems, not in terms of the program-size complexity of the finite string of the axioms.

This new approach combines in a single number the complexity of the axioms and the rules of inference, and the new complexity κ' is never more than c greater and can sometimes be up to $\approx \log_2 \kappa$ less than the old complexity κ . Thus the incompleteness results given here are never weaker and are sometimes somewhat stronger than the incompleteness results in [16].

In addition, this new approach led me to a short, self-contained proof (presented in Section 9) that it is hard to determine scattered bits of the halting probability Ω . While the general theory developed in my monograph [16] is still necessary to substantiate my thesis that there is randomness in arithmetic, there is now a short-cut to the result on the difficulty of determining scattered bits of Ω .

2. Bit String Complexity

Following CHAITIN [6, 16], a computer C is a partial recursive function that maps a program p (a bit string) into an output $C(p)$, which is also a bit string. C is not given the entire program p immediately. Instead, C must request each bit of p , one bit at a time [6]. If a bit is requested it is always provided, so that in a sense programs are initial segments of infinite bit strings. The blank-endmarker approach in [3, 4] may also be considered to request one bit at a time, but differs from the self-delimiting approach used here because requesting a bit may also yield

a blank, indicating that the program has ended.

A more abstract formulation of this self-delimiting program approach, but one that can be proved [6] to be entirely equivalent, is to stipulate that a computer C has the property that no extension of a valid program is a valid program. I.e., if p' is an extension of p , then $C(p')$ cannot be defined if $C(p)$ is defined. In other words, the set of valid programs, which is the domain of definition of the partial recursive function C , is a so-called “prefix-free set.”

The complexity $H_C(x)$ of the string x based on the computer C is the size in bits $|p|$ of the smallest program p for computing x with C :

$$H_C(x) = \min_{C(p)=x} |p|.$$

In addition to this complexity measure, there are related probabilities that take into account **all** programs that produce a given result, not just the smallest ones. The probability $P_C(x)$ of the string x based on the computer C is the probability that C computes x if each bit of the program p is the result of an independent toss of a fair coin:

$$P_C(x) = \sum_{C(p)=x} 2^{-|p|}.$$

It is easy to see that this sum converges and must be between zero and one, because the p that are summed are a prefix-free set. I.e., if a program p is included in this sum, then no extension of p is included in this sum.

Define a universal computer U as follows:

$$U(\overbrace{000 \cdots 000}^{i \text{ 0's}} 1p) = C_i(p).$$

Here C_i is the computer with GÖDEL number i , i.e., the i th computer. Hence

$$H_U(x) \leq H_{C_i}(x) + (i + 1)$$

and

$$P_U(x) \geq P_{C_i}(x) 2^{-(i+1)}$$

for all strings x . The general definition of a universal computer U is that it has the property that for each computer C there is a prefix σ_C such that

$$U(\sigma_C p) = C(p)$$

for all p . I.e., the prefix σ_C tells U how to simulate C . Hence for each computer C there is a constant $\text{sim}_C = |\sigma_C|$ (the cost in bits of **simulating** C) such that

$$H_U(x) \leq H_C(x) + \text{sim}_C$$

and

$$P_U(x) \geq P_C(x) 2^{-\text{sim}_C}$$

for all x . The universal computer U we defined above satisfies this definition with $\text{sim}_{C_i} = i+1$. We pick this particular universal computer U as our standard one and define the complexity $H(x)$ to be $H_U(x)$, and the algorithmic probability $P(x)$ to be $P_U(x)$:

$$H(x) = H_U(x), \quad P(x) = P_U(x).$$

The halting probability Ω is the total algorithmic probability:

$$\Omega = \sum_x P(x) = \sum_{U(p) \text{ is defined}} 2^{-|p|}.$$

Ω is a real number between zero and one, and we also think of it as the infinite bit string of its binary digits. In [16] it is shown that Ω satisfies three different but equivalent definitions of randomness: the constructive measure-theoretic definitions of MARTIN-LÖF and SOLOVAY, and the complexity-theoretic definition of CHAITIN. (An alternative:

$$0 < \Omega' = \sum_x 2^{-H(x)} < 1.$$

This often works just as well.)

3. Discussion

A fundamental theorem shows that the complexity measure H and the algorithmic probability P are closely related:

$$H(x) = -\log_2 P(x) + O(1).$$

Another basic fact is that most bit strings s of length n have close to the maximum possible complexity

$$\max_{|s|=n} H(s) = n + H(n) + O(1) = n + O(\log n).$$

How close are the complexities of most n -bit strings s to the maximum possible? The number of n -bit strings s with complexity k less than the maximum drops off exponentially:

$$\# \left\{ |s| = n : H(s) < n + H(n) - k \right\} < 2^{n-k+c}.$$

The reason for picking a computer U with self-delimiting programs to measure program-size complexity, is that self-delimiting programs can be concatenated. For example, if $U(p) = x$ and $U(q) = y$, then we can compute x and y if we are given pq , the concatenation of the programs p and q .

4. Exhibiting Complex Strings

Now to metamathematics! First we do things the old way.

Following CHAITIN [3, 8, 16], the rules of inference F are a recursively enumerable set of ordered pairs of the form $\langle A, T \rangle$ indicating that the theorem T follows from the axiom A :

$$F = \{ \langle A_1, T_1 \rangle, \langle A_2, T_2 \rangle, \langle A_3, T_3 \rangle, \dots \}.$$

Instead of $\langle A, T \rangle \in F$, one often writes $A \vdash_F T$. (The axiom A is represented as a bit string via some standard binary encoding.) F is fixed, and A varies.

Theorem A (CHAITIN [8]¹): *Consider a formal system F_A consisting of all theorems derived from an axiom A by applying the rules of inference F . The formal system F_A cannot exhibit a specific string with complexity $> H(A) + c_F$. More precisely, if $A \vdash_F H(s) > n$ only if $H(s) > n$, then $A \vdash_F H(s) > n$ only if $n < H(A) + c_F$.*

¹See [1–5] for early versions of this information-theoretic incompleteness theorem.

Proof: Consider a special-purpose computer C that does the following when given a minimal-size program p_k for the natural number k followed by a minimal-size program p_A for the axiom A :

$$C(p_k p_A) = \begin{cases} \text{The first specific string } s^* \\ \text{that can be shown in } F_A \text{ to} \\ \text{have complexity } > k + |p_A|. \end{cases}$$

Here

$$U(p_k) = k, \quad |p_k| = H(k),$$

and

$$U(p_A) = A, \quad |p_A| = H(A).$$

How does C accomplish this? First C simulates running p_k on U to determine the natural number k . Then C simulates running p_A on U to determine the axiom A . Now C knows A and k . C then searches through all proofs derived from the axiom A , searching through the possible proofs in size order, and among those of the same size, in some arbitrary alphabetical order, applying the proof-checking algorithm associated with the fixed rules of inference F to each proof in turn. (More abstractly, C enumerates the set of theorems

$$F_A = \{T : A \vdash_F T\} = \{T : \langle A, T \rangle \in F\}.$$

) In this manner C determines each theorem that follows from the axiom A . C examines each of these theorems until it finds the first one of the form

$$“H(s^*) > j”$$

that asserts that a specific bit string s^* has complexity greater than a specific natural number j that is greater than or equal to k plus the complexity $|p_A|$ of the axiom A :

$$j \geq k + |p_A|.$$

C then outputs the string s^* and halts. Hence

$$H_C(s^*) \leq |p_k p_A|,$$

and

$$k + |p_A| < H(s^*) \leq |p_k p_A| + \text{sim}_C.$$

We therefore have the following crucial inequality:

$$k + H(A) < H(s^*) \leq H(k) + H(A) + \text{sim}_C.$$

This implies

$$k < H(k) + \text{sim}_C = O(\log k),$$

which can only be true for finitely many values of k . Pick c_F to be a k that violates this inequality. It follows that s^* cannot exist for $k = c_F$. The theorem is therefore proved. **Q.E.D.**

5. Set Enumeration Complexity

Following CHAITIN [7, 8], we extend the formalism of Section 2 from finite computations with a single output to infinite computations with an infinite amount of output. Consider a new class of computers, computers that never halt, and which we shall refer to as enumeration computers, or e-computers for short. An e-computer C is given by a total recursive function that maps its program p into the recursively-enumerable set of bit strings $C(p)$. C must request each bit of the program p , and cannot run off the end of p , so p is self-delimiting. But p 's total extent now only emerges in the limit of infinite time.

The complexity $H_C(S)$ of the set S based on the e-computer C is the size in bits $|p|$ of the smallest program p for enumerating S with C :

$$H_C(S) = \min_{C(p)=S} |p|.$$

The probability $P_C(S)$ of the set S based on the e-computer C is the probability that C enumerates S if each bit of the program p is produced by an independent toss of a fair coin:

$$P_C(S) = \sum_{C(p)=S} 2^{-|p|}.$$

Define a universal e-computer U_e as follows:

$$U_e(\overbrace{000 \cdots 000}^{i \text{ 0's}} 1p) = C_i(p).$$

Here C_i is the e-computer with GÖDEL number i , i.e., the i th e-computer. Hence

$$H_{U_e}(S) \leq H_{C_i}(S) + (i + 1)$$

and

$$P_{U_e}(S) \geq P_{C_i}(S) 2^{-(i+1)}$$

for all sets S . The general definition of a universal e-computer U_e is that it has the property that for each e-computer C there is a prefix σ_C such that

$$U_e(\sigma_C p) = C(p)$$

for all p . I.e., the prefix σ_C tells U_e how to simulate C . Hence for each e-computer C there is a constant $\text{sim}_C = |\sigma_C|$ (the cost in bits of **simulating** C) such that

$$H_{U_e}(S) \leq H_C(S) + \text{sim}_C$$

and

$$P_{U_e}(S) \geq P_C(S) 2^{-\text{sim}_C}$$

for all sets S . The universal e-computer U_e we defined above satisfies this definition with $\text{sim}_{C_i} = i + 1$. We pick this particular universal e-computer U_e as our standard one and define the e-complexity² $H_e(S)$ to be $H_{U_e}(S)$, and the enumeration probability $P_e(S)$ to be $P_{U_e}(S)$:

$$H_e(S) = H_{U_e}(S), \quad P_e(S) = P_{U_e}(S).$$

In summary, the e-complexity $H_e(S)$ of a recursively-enumerable set S is the size in bits $|p|$ of the smallest computer program p that makes our standard universal e-computer U_e enumerate the set S . $P_e(S)$ is the probability that our standard universal e-computer U_e enumerates the set S if each bit of the program p is produced by an independent toss of a fair coin.

6. Discussion

The programs p_A and p_B for enumerating two sets A and B can be concatenated. More precisely, the bits in the two programs p_A and p_B

²In full, the “enumeration complexity.”

can be **intertwined** or merged in the order that they are read by two copies of the universal e-computer U_e running in parallel and sharing a single program bit stream. Thus e-complexity is additive, because the size of the intertwined bit string $p_A \oplus p_B$ is the sum of the sizes of the original strings p_A and p_B .

Let's show some applications of intertwining. Define the e-complexity of a function f to be the e-complexity of the graph of f , which is the set of all ordered pairs $\langle x, f(x) \rangle$. By intertwining,

$$H_e(\{f(x) : x \in X\}) < H_e(f) + H_e(X) + c.$$

Here is the cartesian product of two sets:

$$H_e(\{\langle x, y \rangle : x \in A, y \in B\}) < H_e(A) + H_e(B) + c.$$

Two other examples of intertwining:

$$H_e(A \cap B) < H_e(A) + H_e(B) + c,$$

and

$$H_e(A \cup B) < H_e(A) + H_e(B) + c.$$

Here is a horse of a different color:

$$H(\varphi(\psi(x))) < H_e(\varphi) + H_e(\psi) + H(x) + c.$$

7. Exhibiting Complex Objects

While a minimal-size program for the computer U tells us its size as well as its output

$$H(x, H(x)) = H(x) + O(1),$$

this is not the case with a minimal-size program for the e-computer U_e . Instead we only get its size in the limit from below:

$$H_e(X, \{0, 1, 2, \dots, H_e(X)\}) = H_e(X) + O(1).$$

(It is annoying to have to define H and H_e for multiple arguments. Intuitively, one simply computes several outputs simultaneously.³) In the proof that

$$A \vdash_F H(s) > n \implies n < H(A) + c_F$$

in Section 4, it is important that C knows $|p_A| = H(A)$ as well as A . So it looks like we cannot prove that

$$“H(s) > n” \in T \implies n < H_e(T) + c.$$

Surprisingly, everything works anyway.

Theorem B: *Consider a formal system consisting of a recursively enumerable set T of theorems. The formal system T cannot exhibit a specific string with complexity $> H_e(T) + c$. More precisely, if a theorem of the form “ $H(s) > n$ ” is in T only if it is true, then “ $H(s) > n$ ” is in T only if $n < H_e(T) + c$.*

Proof: Consider a special-purpose computer C that is given as its program a minimal-size program p_k for the singleton set $\{k\}$ of the natural number k appropriately intertwined with a minimal-size program p_T for the set of theorems T . Here

$$U_e(p_k) = \{k\}, \quad |p_k| = H_e(\{k\}),$$

and

$$U_e(p_T) = T, \quad |p_T| = H_e(T).$$

When C is given p_k intertwined with p_T it does the following. C runs p_k and p_T in parallel on U_e to determine k and enumerate the set of theorems T . As C enumerates T , C keeps track of the number ρ of bits of p_T that it has read. At some point C will find k . Thereafter, C continually checks T until C finds a theorem of the form

$$“H(s^*) > j”$$

³Here are some more formal definitions. For H , one can compute a tuple $\langle x, y, z, \dots \rangle$. The tuple mapping is a computable one-to-one correspondence between bit strings and singletons, pairs, triples, ... of bit strings. For H_e , one can enumerate several sets A, B, C, \dots by prefixing the elements of each with a different prefix:

$$\{1x : x \in A\} \cup \{01x : x \in B\} \cup \{001x : x \in C\} \cup \dots$$

asserting that a specific string s^* has complexity greater than a specific natural number j that is greater than or equal to k plus the current value of ρ :

$$j \geq k + \rho.$$

C then outputs the string s^* and halts. It is possible that not all bits of p_k and p_T have been read by C ; unread bits of p_k and p_T are not actually included in the intertwined program for C . Let p'_k and p'_T be the portions of p_k and p_T that are actually read. Thus the final value of ρ must satisfy

$$\rho = |p'_T| \leq H_e(T).$$

In summary,

$$C(p'_k \oplus p'_T) = s^*, \quad "H(s^*) > j" \in T, \quad j \geq k + \rho.$$

Thus s^* must have the property that

$$H_C(s^*) \leq |p'_k| + |p'_T| \leq H_e(\{k\}) + \rho.$$

We therefore have the following crucial inequality:

$$k + \rho < H(s^*) \leq H_e(\{k\}) + \rho + \text{sim}_C.$$

Note that

$$H_e(\{k\}) + \rho + \text{sim}_C < O(\log k) + \rho.$$

Hence

$$k + \rho < H(s^*) < O(\log k) + \rho.$$

This implies

$$k < O(\log k),$$

which can only be true for finitely many values of k . Pick c to be a value of k that violates this inequality. For $k = c$, s^* cannot exist and C can never halt. Thus T cannot contain any theorem of the form

$$"H(s^*) > j"$$

with

$$j \geq H_e(T) + k,$$

because

$$H_e(T) + k \geq k + \rho$$

and C would halt. The theorem is therefore proved. **Q.E.D.**

Recall the setup in Theorem A: the fixed rules of inference F and the variable axiom A yield the set of theorems F_A . Let's apply Theorem B to the set of theorems F_A so that we can compare how powerful Theorems A and B are. Here is what we get:

$$\begin{aligned} \text{Theorem A: } & A \vdash_F H(s) > n \implies n < H(A) + c_F. \\ \text{Theorem B: } & "H(s) > n" \in F_A \implies n < H_e(F_A) + c. \end{aligned}$$

The e-complexity $H_e(F_A)$ is never more than c bits larger and can sometimes be up to $\approx \log_2 H(A)$ bits smaller than the complexity $H(A)$. This is because it is sometimes much easier to give the size $|p|$ of a program in the limit from below than to give the size of a program and then halt. It all boils down to the fact that $H_e(\{0, 1, 2, \dots, |p|\})$ can be insignificant in comparison with $H(|p|)$ (see [7, Section 3]). Thus Theorem B is never weaker and sometimes is a little stronger than Theorem A.

Let's look at some other consequences of the method used to establish Theorem B.

We have seen that one can't exhibit complex strings. What about sets? One can't exhibit e-complex sets:

$$"H_e(U_e(p)) > n" \in T \implies n < H_e(T) + c.$$

Here is a bound on what can be accomplished with a single axiom A and the rules of inference F :

$$A \vdash_F H(s) > n \implies n < H(A) + H_e(F) + c.$$

Recall that Theorem A only asserted that

$$A \vdash_F H(s) > n \implies n < H(A) + c_F.$$

Consider the set of theorems T derived from a set of axioms A using the rules of inference F . (Now F is a recursively enumerable

set of ordered pairs each consisting of a finite set of axioms and a consequence.) We have

$$H_e(T) \leq H_e(A) + H_e(F) + c.$$

And we get the following bound on what can be accomplished with the set of axioms A and the rules of inference F :

$$A \vdash_F H(s) > n \implies n < H_e(A) + H_e(F) + c.$$

8. Determining Bits of Ω

In the same spirit as Theorem B in Section 7, here is a new and improved version of the key theorem in [16, Chapter 8] that one can determine at most $n + c$ bits of the halting probability Ω with a formal system of complexity n .

Theorem C: *Consider a formal system consisting of a recursively enumerable set T of theorems. If the formal system T has the property that a theorem of the form*

“The n th bit of Ω is 0/1.”

is in T only if it is true, then at most $H_e(T) + c$ theorems of this form are in T . In other words, if the e-complexity of T is n , then T can enable us to determine the positions and values of at most $n + c$ scattered bits of Ω .

Proof: (By reductio ad absurdum.) Suppose on the contrary that for each k there is a formal system T that enables us to determine $H_e(T) + k$ bits of Ω . We shall show that this contradicts the fact (see [16]) that Ω is a MARTIN-LÖF random real number.

Here is a way to produce a set of intervals A_k that covers Ω and has measure $\mu(A_k) \leq 2^{-k}$. I.e., a way that given k , we can enumerate a set of intervals A_k that includes Ω and whose total length is $\leq 2^{-k}$.

Start running for more and more time all possible programs p on the standard universal e-computer U_e . If at any point we have read $\rho \leq |p|$ bits of a program p while enumerating its output $U_e(p)$ and this output includes $\rho + k$ theorems of the form

“The n th bit of Ω is 0/1.”

determining $\rho + k$ bits of Ω , then we do the following:

1. The $\rho + k$ theorems in $U_e(p)$ give us a set of intervals of total measure $2^{-(\rho+k)}$ that covers Ω . More precisely, this set of intervals covers Ω if $U_e(p)$ is a truthful formal system. We add these intervals to the set of intervals A_k .
2. We stop exploring this subtree of the tree of all possible programs p . In other words, we don't continue with the current program p nor do we examine any program whose first ρ bits are the same as the first ρ bits of p . This removes from further consideration all programs in an interval of length $2^{-\rho}$, i.e., a set of programs of measure $2^{-\rho}$.

For each k , the set of intervals A_k will have total measure $\mu(A_k) \leq 2^k$. Ω cannot be in all the A_k or Ω would not be MARTIN-LÖF random, which it most certainly is [16]. Therefore Ω is in only finitely many of the A_k . Let c be the first k for which Ω is not in A_k , i.e., such that we never find a way of determining $\rho + k$ bits of Ω with only ρ bits of axioms. **Q.E.D.**

9. A Different Proof

In Section 8 it was shown that a formal system of e-complexity n cannot determine the positions and values of more than $n+c$ bits of Ω (Theorem C). The proof is like an iceberg, because it depends on the theory systematically developed in the second half of my monograph [16]. Here is a rather different proof that is essentially self-contained.⁴

Theorem C: *A formal system T can enable us to determine the positions and values of at most $H_e(T) + c$ bits of Ω . In other words, if a theorem of the form*

“The n th bit of Ω is 0/1.”

⁴An analogous situation occurs in elementary number theory. See the remarkably simple ZERMELO–HASSE–LORD CHERWELL proof of the unique prime factorization theorem in RADEMACHER and TOEPLITZ [17, p. 200].

is in T only if it is true, then at most $H_e(T) + c$ theorems of this form are in T .

Proof #2: Consider the following special-purpose computer C . Given a program

$$\overbrace{000 \cdots 0001}^{k \text{ bits}} p_T x,$$

C does the following. First C reads the initial run of 0 bits and the 1 bit at its end. The total number of bits read is k . Then C starts running on U_e the remainder of its program, which begins with a minimal-size program p_T for enumerating T :

$$U_e(p_T) = T, \quad |p_T| = H_e(T).$$

As C enumerates T , C counts the number of bits

$$\rho \leq H_e(T)$$

of p_T that it has read. The moment that C has enumerated enough theorems in the formal system T to find the values and positions of $\rho + 2k$ bits of Ω , C stops enumerating the theorems of the formal system T . (Unread bits of p_T are not actually included in the program for C . Let p'_T be the portion of p_T that is actually read. Thus the final value of ρ equals $|p'_T|$ and may actually be **less than** $H_e(T)$ if not all bits of the minimal-size program for enumerating T are needed.) Now C knows $\rho + 2k$ bits of Ω . Next C determines the position n of the bit of Ω that it knows that is farthest from the decimal point. In other words, C finds the largest n in the first $\rho + 2k$ theorems of the form

“The n th bit of Ω is 0/1.”

in T , where $\rho = |p'_T|$ is the number of bits of p_T that are read. Consider the string Ω_n of the first n bits of Ω :

$$\Omega_n = \beta_1 \beta_2 \beta_3 \cdots \beta_n.$$

From T , C has determined β_n and $\rho + 2k - 1$ other bits of Ω_n . To fill in the gaps, the remaining $n - \rho - 2k$ bits of Ω_n are provided to C as the remainder of its program, x , which is exactly $n - \rho - 2k$ bits long.

(For C 's program to be self-delimiting, it is crucial that at this point C already knows exactly how long x is.) Now C knows the first n bits of Ω , and it outputs them and halts:

$$C(\overbrace{000 \cdots 0001}^{k \text{ bits}} p'_T x) = \Omega_n.$$

Note that the size of C 's program is

$$|\overbrace{000 \cdots 0001}^{k \text{ bits}}| + |p'_T| + |x| = k + \rho + (n - \rho - 2k).$$

This of course simplifies as follows:

$$k + \rho + (n - \rho - 2k) = n - k.$$

Hence

$$H_C(\Omega_n) \leq n - k.$$

We therefore have the following crucial inequality:⁵

$$n - c' < H(\Omega_n) \leq n - k + \text{sim}_C.$$

Hence

$$k < c' + \text{sim}_C.$$

Taking

$$k = c' + \text{sim}_C$$

we get a contradiction. Thus T cannot yield

$$\rho + 2k = \rho + 2(c' + \text{sim}_C) \leq H_e(T) + 2(c' + \text{sim}_C)$$

bits of Ω . The theorem is proved with

$$c = 2(c' + \text{sim}_C).$$

Q.E.D.

⁵It is easy to see that there is a computer C' such that

$$U(p) = \Omega_n \implies C'(p) = \text{the first string } x \text{ with } H(x) > n.$$

Hence

$$H(\Omega_n) > n - \text{sim}_{C'} = n - c'.$$

I.e., Ω is a CHAITIN random real. For the details, see the chapter "Chaitin's Omega" in GARDNER [29], or the proof that Ω is CHAITIN random in [16].

10. Diophantine Equations

Now let's convert Theorem C into an incompleteness theorem about diophantine equations.⁶

We arithmetize Ω in two diophantine equations: one polynomial, the other exponential. Ω can be obtained as the limit of a computable sequence of rational numbers Ω_l :

$$\Omega = \lim_{l \rightarrow \infty} \Omega_l.$$

(Careful: in Section 9, Ω_l meant something else.) For example, let Ω_l be the sum of $2^{-|p|}$ taken over all programs p of size $\leq l$ that halt in time $\leq l$:

$$\Omega_l = \sum_{\substack{|p| \leq l \\ U(p) \text{ halts in time } \leq l}} 2^{-|p|}.$$

The methods of JONES and MATIJASEVIČ [10, 22, 31] enable one to construct the following:

1. A diophantine equation

$$P(k, l, x_1, x_2, x_3, \dots) = 0$$

that has one or more solutions if the k th bit of Ω_l is a 1, and that has no solutions if the k th bit of Ω_l is a 0.

2. An exponential diophantine equation

$$L(k, l, x_2, x_3, \dots) = R(k, l, x_2, x_3, \dots)$$

that has exactly one solution if the k th bit of Ω_l is a 1, and that has no solutions if the k th bit of Ω_l is a 0.

Since in the limit of large l the k th bit of Ω_l becomes and remains correct, i.e., identical with the k th bit of Ω , it follows immediately that:

⁶My first information-theoretic incompleteness theorem about diophantine equations is stated without proof in the introduction of my 1974 paper [4]. A better one is mentioned in my 1982 paper [9, Section 4]. Finally, two papers [11, 12] and a book [16] give number theory their undivided attention.

1. There are infinitely many values of l for which the diophantine equation

$$P(k, l, x_1, x_2, x_3, \dots) = 0$$

has a solution iff the k th bit of Ω is a 1.

2. The exponential diophantine equation

$$L(k, x_1, x_2, x_3, \dots) = R(k, x_1, x_2, x_3, \dots)$$

has infinitely many solutions iff the k th bit of Ω is a 1.

Consider the following questions:

1. For a given value of k , are there infinitely many values of l for which the diophantine equation

$$P(k, l, x_1, x_2, x_3, \dots) = 0$$

has a solution?

2. For a given value of k , does the exponential diophantine equation

$$L(k, x_1, x_2, x_3, \dots) = R(k, x_1, x_2, x_3, \dots)$$

have infinitely many solutions?

By Theorem C, to answer any n cases of the first question or any n cases of the second question requires a formal system of e-complexity $> n - c$.

For discussions of the significance of these information-theoretic incompleteness theorems, see [13, 15, 18–21, 23–30].

References

- [1] G. J. CHAITIN, “Computational complexity and Gödel’s incompleteness theorem,” *AMS Notices* **17** (1970), 672.
- [2] G. J. CHAITIN, “Computational complexity and Gödel’s incompleteness theorem,” *ACM SIGACT News* **9** (1971), 11–12.

- [3] G. J. CHAITIN, “Information-theoretic computational complexity,” *IEEE Transactions on Information Theory* **IT-20** (1974), 10–15.
- [4] G. J. CHAITIN, “Information-theoretic limitations of formal systems,” *Journal of the ACM* **21** (1974), 403–424.
- [5] G. J. CHAITIN, “Randomness and mathematical proof,” *Scientific American* **232**:5 (1975), 47–52.
- [6] G. J. CHAITIN, “A theory of program size formally identical to information theory,” *Journal of the ACM* **22** (1975), 329–340.
- [7] G. J. CHAITIN, “Algorithmic entropy of sets,” *Computers & Mathematics with Applications* **2** (1976), 233–245.
- [8] G. J. CHAITIN, “Algorithmic information theory,” *IBM Journal of Research and Development* **21** (1977), 350–359, 496.
- [9] G. J. CHAITIN, “Gödel’s theorem and information,” *International Journal of Theoretical Physics* **22** (1982), 941–954.
- [10] J. P. JONES and Y. V. MATIJASEVIČ, “Register machine proof of the theorem on exponential diophantine representation of enumerable sets,” *Journal of Symbolic Logic* **49** (1984), 818–829.
- [11] G. J. CHAITIN, “Randomness and Gödel’s theorem,” *Mondes en Développement* **54–55** (1986), 125–128.
- [12] G. J. CHAITIN, “Incompleteness theorems for random reals,” *Advances in Applied Mathematics* **8** (1987), 119–146.
- [13] G. J. CHAITIN, “Randomness in arithmetic,” *Scientific American* **259**:1 (1988), 80–85.
- [14] G. J. CHAITIN, *Information, Randomness & Incompleteness—Papers on Algorithmic Information Theory*, 2nd Edition, Singapore: World Scientific (1990).

- [15] G. J. CHAITIN, “Undecidability and randomness in pure mathematics,” (transcript of a lecture delivered 28 September 1989 at a SOLVAY conference in Brussels), in [14, pp. 307–313].
- [16] G. J. CHAITIN, *Algorithmic Information Theory*, 3rd Printing, Cambridge: Cambridge University Press (1990).
- [17] H. RADEMACHER and O. TOEPLITZ, *The Enjoyment of Mathematics*, New York: Dover (1990).
- [18] G. J. CHAITIN, “A random walk in arithmetic,” *New Scientist* **125**:1709 (1990), 44–46. Reprinted in N. HALL, *The New Scientist Guide to Chaos*, Harmondsworth: Penguin (1991).
- [19] J. L. CASTI, *Searching for Certainty*, New York: Morrow (1990).
- [20] G. J. CHAITIN, “Number and randomness,” (transcript of a lecture delivered 15 January 1991 at the Technical University of Vienna). In M. E. CARVALLO, *Nature, Cognition and System*, Vol. 3, Dordrecht: Kluwer (1992), in press.
- [21] G. J. CHAITIN, “Le hasard des nombres,” *La Recherche* **22**:232 (1991), 610–615.
- [22] J. P. JONES and Y. V. MATIJASEVIČ, “Proof of the recursive unsolvability of Hilbert’s tenth problem,” *American Mathematical Monthly* **98** (1991), 689–709.
- [23] D. RUELLE, *Chance and Chaos*, Princeton: Princeton University Press (1991).
- [24] D. RUELLE, *Hasard et Chaos*, Paris: Odile Jacob (1991).
- [25] L. BRISSON and F. W. MEYERSTEIN, *Inventer L’Univers*, Paris: Les Belles Lettres (1991).
- [26] J. A. PAULOS, *Beyond Numeracy*, New York: Knopf (1991).
- [27] J. D. BARROW, *Theories of Everything*, Oxford: Clarendon Press (1991).

- [28] T. NØRRETRANDERS, *Mærk Verden*, Denmark: Gyldendal (1991).
- [29] M. GARDNER, *Fractal Music, Hypercards and More...*, New York: Freeman (1992).
- [30] P. DAVIES, *The Mind of God*, New York: Simon & Schuster (1992).
- [31] C. SMORYŃSKI, *Logical Number Theory I*, Berlin: Springer-Verlag (1991).