

From Complexity to Intelligence

Algorithmic Information Theory and mathematics

Nils Holzenberger

September 28, 2026

Outline

- 1 Computability of C
- 2 Algorithmic probability
 - A different view of probabilities
 - Prefix complexity
 - Randomness
- 3 Computability
 - Chaitin's Ω
 - Algorithmic classes
 - Gödel's theorem

C is not computable :'(

What does *computable* mean?

There exists a Turing machine which takes x as input and output $C(x)$ in finite time.

C is not computable — proof

Assume C is computable, let f be its implementation.

```
def more_complex(n):
    ii = 1
    while True:
        s = bin(ii)[2:] # binary encoding of ii
        if f(s) > n:
            return s
        ii += 1
```

Is `more_complex` well-defined? Does it terminate? Yes because there are a finite number of binary strings of complexity $\leq n$ (in fact, fewer than 2^{n+1} , see chapter on compression).

What does `more_complex` do? Input is an integer n , output is a binary string s such that $C(s) > n$.

C is not computable — proof

I can use `more_complex` to generate s from n . So

$$C(s) \leq C(\text{more_complex}) + C(n) + O(1)$$

By definition of s : $C(s) \geq n$

$$\text{So } \forall n \in \mathbb{N} \quad n \leq C(n) + O(1)$$

But $C(n) \leq O(1) + \log(n)$ (The machine I currently consider can simulate another Turing machine which implements the standard binary encoding of integers).

We get $\forall n \in \mathbb{N} \quad n \leq O(1) + \log(n)$ *quod est absurdum*.

C is not computable

...and yet it will serve to define other concepts and objects in mathematics.

When doing so, we move the object or concept becomes part of (theoretical) computer science.

Probabilities

Regular probabilities:

- Drawing balls in urns
- Sets and measures
- Monotonicity w.r.t. inclusion

AIT's probabilities:

- Drawing binary strings
- Generating things from them

You meet Paul, your friend from high school, at a class reunion. You haven't seen Paul in 10 years. You remember that he used to be an outspoken advocate for fighting climate change. You see him arrive at the reunion in a car. What is most likely:

- *Paul is driving a SUV*
- *Paul is driving an electric SUV*

Intuition

Objects of low complexity are more probable than objects of high complexity.

Initial definition:

- We can draw binary sequences, each bit has probability $\frac{1}{2}$ to be 0, $\frac{1}{2}$ to be 1.
- The probability of an object is the probability of drawing a binary sequence that generates this object.

$$p(s) = \sum_{U(p)=s} 2^{-|p|} \approx 2^{-C(s)}$$

Initial definition

$$p(s) = 2^{-C(s)}$$

Consider integers with binary encoding. $p(n) = 2^{-\log(n)} = \frac{1}{n}$.

$$\sum_n p(n) = \sum_n \frac{1}{n} = \dots$$

Another problem: I am drawing unbounded binary sequences, without a stopping criterion.

Prefix complexity

Huffman vs. Shannon encoding

Shannon

Symbol	Code
A	0
B	1
C	10

Huffman

Symbol	Code
A	0
B	10
C	11

010010 = ?

Shannon: ABAABA? ACAC? ...

Huffman: ABAB.

Huffman encoding is a prefix code

Prefix encoding

- Double each bit except the last one
- The last bit is encoded as 01 if 0, and 10 if 1
- Examples:
 - $011100 \rightarrow 001111110001$
 - $0111 \rightarrow 00111110$
- Any non-prefix code can be turned into a prefix code; any ordinary Turing machine can be turned into a prefix Turing machine

Prefix code for integers

$$K(n) = 2\log(n)$$

$$K(n) = 2\log(\log(n)) + \log(n)$$

$$K(n) = 2\log(\log(\log(n))) + \log(\log(n)) + \log(n)$$

...

$$K(n) = \log(n) + \alpha$$

Algorithmic probability

$p_U(s) = \sum_{U(p)=s} 2^{-|p|}$ where U is a prefix Turing machine

p_U is well defined:

- Think of the binary tree involved in constructing a Huffman code
- Or, consider the intervals $[0.p, 0.p + 2^{-|p|}]$

Prefix complexity

$$K_M(x) = \min_p \{\ell(p) \mid M(p) = x\}$$

From now on:

- C = plain complexity
- K = prefix complexity

⚠ On the Internet, people may use either notation.

Chain rule again

$$K(x, y) = K(x) + K(y|x) + O(\log(K(x, y)))$$

$$K(x, y) = K(x) + K(y|p(x)) + O(1)$$

Random numbers

What is a random number? Are some numbers more random than others?
Is it a continuous or a discrete property?

We will consider infinitely long binary sequences.

Richard von Mises

A binary sequence is random if, for any rule that selects subsequences independently of their content, the proportion of 1's in the subsequence is 50%. Richard von Mises, 1910

Objection: Champernowne's constant

Emile Borel

Assume there exists a definition of randomness. Let n be the smallest such number. Then n is not random. Emile Borel, 1927

Per Martin-Löf

Per Martin-Löf's definition: see lab session

Algorithmic Information Theory

A sequence is random if it is not very compressible. Andrei Kolmogorov (1965), Gregory Chaitin (1975), Leonid Levin (1975)

$$\exists c, \forall m, K(x_m) > m - c$$

where x_m are the first m elements in the sequence x .

- Is a finite binary sequence random?
- Is π random?
- Are there any mathematical objects which are random?

Chaitin's Ω

$$\Omega = \sum_{p \text{ s.t. } M(p) \text{ halts}} 2^{-\ell(p)}$$

Some facts about Ω :

- Ω is not computable
- Ω can be approximated by a computable sequence, but the error cannot be bounded
- Let S be a formal system such that propositions of the form “ ω is a prefix of Ω ” are true. Such an assertion can only be proven if $\ell(w) \leq K(S) + c$.

Mathematical categories

- Integers $\mathbb{N}$, $\mathbb{Z}$
- Rationals $\mathbb{Q}$
- Real numbers $\mathbb{R}$
- Complex numbers $\mathbb{C}$

Recursive or decidable set

Definition There exists a Turing machine which can determine in finite time whether $x \in A$

Examples

- Prime numbers
- The set of solutions to $n + m = p$

Recursively enumerable set

Definition There exists a Turing machine which halts if and only if the input belongs to A

Examples Theorems in first-order logic provable by resolution.

Approximable set

Definition Finite or contains a subset that is finitely enumerable

Example The set of true arithmetical formulas

Counter-example True formulas of the form $C(s) = n$ (s a finite binary sequence).

Incompressible set

Definition Let S be a sound formal system to check whether x belongs to A . S can generate at most $K(S) + c$ items in the set.

Example The digits of Ω .

Classes of numbers

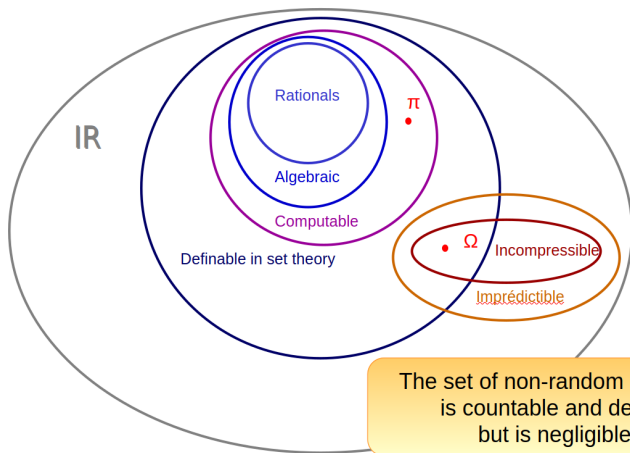
Mathematical numbers:

- Integers, Rational numbers
- Algebraic numbers
- Transcendental number
- Normal numbers (each digit's frequency is b^{-k})
- Unpredictable (impossible to win by gambling on the next digit)

AIT numbers:

- Calculable
- Random (or incompressible)

Numbers



The set of non-random numbers is countable and dense, but is negligible!

Two proofs

Kurt Gödel, 1931: *some mathematical propositions cannot be proven nor disproven*. The proof is some 70 pages long.

(Kleene, 1943: proof showing equivalence to halting problem)

Gregory Chaitin, 1970: There exists $m_0 \in \mathbb{N}$ such that $\forall m \geq m_0$ there exist statements of the form $K(n) > m$ that are true but not provable.

Gregory Chaitin's proof

1. There are at most 2^{m+1} integers n such that $K(n) \leq m$, so there must be an infinite number of true statements of the form $K(n) > m$.
2. Let $m \in \mathbb{N}$ such that there exists an n where $K(n) > m$ is provable.

I can find n by enumerating all theorems and knowing m , so
 $K(n) \leq 2\log(m) + O(1)$

By definition $m < K(n)$ so $m \leq 2\log(m) + O(1)$ so $m \leq m_0$.

Contraposition: $\forall m, m > m_0 \Rightarrow \exists n, [K(n) > m]$ is not provable

Gödel's theorem extended

Let S be a formal system. (For example arithmetics and its axioms)

S can only describe propositions of the type $K(x) = n$ if $K(S) + c > n$

S cannot create more information than it already contains

Conclusion

Complexity is not computable. But it can still be used to:

- Define randomness
- Define probabilities
- Define classes of sets and numbers
- Prove Gödel's theorem