Algorithmic Information Theory: The Basics

Adam Elga[*]

IAP January 10, 2000

1 Preliminaries

1.1 Turing machines

Turing machine An idealized computing device attached to a *tape*, each square of which is capable of holding a symbol. We write a *program* p (a finite binary string) on the tape, and start the machine. If the machine halts with string o written at a designated place on the tape, then o is the *output*.

Universal Turing machine A Turing machine capable of simulating any other Turing machine, in the following sense. Given an input that encodes the command "Simulate machine k on program p", a universal machine will output whatever machine k would output if given program p.

1.2 Definition of complexity

The *complexity of string s relative to machine T* is defined to be the length of the shortest program that gets T to produce s as output.

We choose some universal Turing machine U, and define the *complexity of string s* to be the complexity of s relative to U.

How much does the choice of universal machine matter?

[*] `adam@philosophers.net`

Lots. For any string s, there is a universal machine U_s such that the complexity of s relative to U_s is zero.

Not much. Suppose that U and U' are both universal machines. Then there is a constant k such that for any s, the complexity of s relative to U never differs from the complexity of s relative to U' by more than k. So for increasingly long strings, the difference between U and U' becomes less and less significant.

Upshot: this complexity measure allows us to bootstrap a single qualitative simplicity judgment into many quantitative simplicity judgments.

2 Basic facts

1. Strings never have complexities much greater than their lengths. There is a constant k such that no string's complexity is more than k greater than its length.
2. Low-complexity strings are relatively rare. Example: of the strings of length 1000, only a tiny fraction have complexity less than 900.
3. A minimal program has a complexity approximately equal to its length.
4. The complexity function $C(\cdot)$ is not computable.

   Hint on how to prove this: Berry paradox.

   D: $\boxed{\text{The smallest number not describable in fewer than twenty syllables.}}$

   D describes some number, since there are only finitely many under-twenty-syllable descriptions.

   But suppose that **D** refers to n. Then n is describable in nineteen syllables, so **D** doesn't refer to n. Contradiction.

3 Randomness

Two approaches to the question of whether a string is random:

1. Consider what sort of process *produced* the string. Call the string *process-random* if it was produced by the right sort of chancy process.

2. Pay no attention to what produced the string. Instead, call the string *product-random* if it is appropriately unpatterned.

We'll focus on the second approach, as applied to infinite binary strings.

Think of a minimal-length program for a finite string as a compressed version of the string. The *compressibility* of a finite string s is defined to be the length of s minus the length of a minimal-length program for s.

Intuition: an infinite string that is product-random ought to have initial segments that aren't very compressible. That motivates the following definition:

An infinite string is *product-random* if and only if there is an upper bound B such that no initial segment of the string has compressibility greater than B.

Tweak: require programs to be *self-delimiting.*

In the long run, product-random sequences have just as many `1`s as `0`s. Also, product-random sequences cannot be exploited by gambling machines.

4 Incompleteness

The *language of arithmetic* contains logical symbols, expressions that denote numbers, and the addition and multiplication signs.

In the language of arithmetic, one can make numerical assertions such as:

- For all x, for all y, $x + y = y + x$.
- There exists an x such that for all y, $x \times y = y \times y$.

Suppose that we've got some *axioms* that are such that some Turing machine prints the axioms out in order.

Suppose that we've got some *rules of inference* for deriving consequences of the axioms, rules such that there is a machine M that prints out every sentence derivable from the axioms using the rules (and prints no other sentences).

Question: Can it be that our axioms and rules satisfy both of the following conditions?

1. Every sentence derivable from the axioms on the basis of the rules is true. (In other words, every sentence that M prints is true.)
2. Every true arithmetical sentence is derivable from the axioms on the basis of the rules. (In other words, M eventually prints every true arithmetical sentence.)

References

[1] Gregory Chaitin. *Algorithmic information theory.* Cambridge University Press, 1987. This book develops algorithmic prefix complexity, emphasizing its analogies with the entropy of classical information theory. Chapter 6 investigates infinite random sequences. Available at http://www.cs.auckland.ac.nz/CDMTCS/chaitin/cup.pdf.

[2] Ming Li and Paul Vitanyi. *An introduction to Kolmogorov complexity and its applications.* Springer, 1997. A comprehensive textbook. This book has everything, including extensive notes on the history of the various ideas.

[3] Michiel van Lambalgen. Von mises' definition of random sequences reconsidered. *Journal of Symbolic Logic*, 4:725–755, 1987. A technical article comparing various criteria for randomness.

[4] Michiel van Lambalgen. Algorithmic information theory. *Journal of Symbolic Logic*, 54(4):1389–1400, 1989. Argues against some of Chaitin's claims about the relationship between algorithmic information theory and Godel's incompleteness theorems. If you read anything by Chaitin, you should also read this article.