Intricacy handout

Adam Elga

February 18, 2003

1 Algorithmic Complexity

A.K.A. Kolmogorov-Chaitin-Solomonov complexity. Motivating idea: simple objects admit of short descriptions. Example: scribbles on the board, vs picture of a house. Example: patterned sequence of digits vs. unpatterned sequence.

The *complexity of string s relative to machine T* is defined to be the length of the shortest program (i.e., input) that gets T to produce s as output, or ∞ if no program gets T to produce s as output.

Choose some universal Turing machine U as our reference machine, and define the *complexity of string s* to be the complexity of s relative to U.

How much does the choice of reference machine matter?

Lots, locally. 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, in the limit. 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. (Why?)

So: fixing U and U', in the limit as the strings get larger, the particular choice of UTM matters less and less.

2 Berry paradox

(This presentation follows that of G. Chaitin.)

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

Notice that there are only finitely many under-twenty-syllable descriptions. Therefore, not every natural number is describable in under twenty syllables. Therefore, there must be a least number not describable in under twenty syllables. Therefore, **D** describes some number. Which one?

Suppose for contradiction that **D** describes n. Then n is describable in nineteen syllables, so **D** doesn't describe n. Contradiction. What has gone wrong?

3 Uncomputability of intricacy

The *intricacy* $i(n)$ of n is the size (# of states in) the smallest machine that outputs n when started on a blank tape.

Let $L(n)$ be the least number that requires a machine with more than n states to produce.

Suppose for contradiction that some Turing machine computes i.

How could you rejig such a machine to get a machine that computes L?

Call the resulting machine T_L, and suppose that it has k states.

Consider $L(1000000k) = V$. V is the least number that requires a machine with more than $1000000k$ states to produce.

But now consider the description "the least number that requires a machine with more than $1000000k$ states to produce." That looks like a compact recipe for producing V. How can you design a machine (with fewer than $1000000k$ states) that produces V according to that recipe?

4 Uncomputability and incompleteness

Can a given domain of discourse be mechanized?

Say that a domain of discourse can be *Turing-mechanized* iff there is some TM that prints out all and only the truths in that domain of discourse.

Equivalently, a domain can be Turing-mechanized iff there is a TM that prints out a set of *axioms* for the domain of discourse, and a Turing-computable rule for deriving theorems from the axioms.

Example: consider any domain of discourse that addresses all claims of the form:

> The intricacy of n is k.

That domain of discourse cannot be Turing-mechanized. (Why?) The same goes for domains that address all claims of the form "Turing machine m halts on input n."

Important application: Arithmetic. In the language of arithmetic, one can make numerical assertions such as:

- It is not the case that for all x, for all y, $x + y = y + y$.
- There exists an x such that for all y, $x \times y = y \times y$.

Claims such as "244 is prime", "there is a consecutive pair of primes that differ by 17", etc. can all be expressed in the language of arithmetic. Less obviously, so can claims about Turing machines.

Punchline (a claim close to Godel's first incompleteness theorem): arithmetic—or any consistent domain of discourse that includes it—cannot be Turing-mechanized.