Algorithmic Information Theory

- We have seen some ways to treat information (flow), starting from the early syntactic approach of Claude Shannon. It was related to technical matters of his day.
- Today another *syntactic* approach is promi-nent: Gregory Chaitin´s Algorithmic Information Theory. It is related to matters of computer programming, i.e. it too is related to technical matters of today.

C*entre for the*
S*tudy of*
L*ogic,*
L*anguage, and*
I*nformation*

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Manuel Bremer

A Theory of Information Content

- Algorithmic Information Theory (AIT) is a theory of information content, not of information flow. It deals with word strings.
- The basic measure is the same like in the original syntactic approach: bits.
- But AIT focuses not simply on the coding scheme but on matters of generating a word string by a program.
- It is related to complexity theory.

Information Content (Outline)

- A string has some measure in bits.
- The information content of a string is the *length of the shortest program (in bits)* which is needed to *generate* the string.
- The length of the shortest program for a string is also its *complexity*.
- (For practical purposes a version of LISP is used to have a working model of AIT.)

Randomness

- A finite string of length n can be "programmed" by having it simply printed (with length n+k, k being the length in bits of the minimal code to print it).
- The real problem are *infinite* strings.
  [What does this tell us about the applicability of AIT? AIT´s most famous result (a random number in Arithmetic) deals with meta-mathematics!]
- A string is *random* if the size of the shortest program for it, if there is any, is not shorter than the string itself.

Randomness (Some Details)

- *Most* strings are random, since there are more strings than well-formed programs.
- There are 2^n strings of length n, and less than 2^{n-k} programs of length less than n-k. Thus the number of strings of length n and complexity less than n-k decreases *exponentially* as k increases.
- So the *great majority* of strings of length n are of complexity very close to n.

Algorithmic Information Content (Definitions and Theorems)

- Definition 1. A computer is a partial recursive function C(p). Its argument p is a binary string. The value of C(p) is the binary string output by C given the program p. If C(p) is undefined the computation does not halt.
- Definition 2. The complexity $I_C(s)$ of a binary string is defined to be the length of the shortest program p that makes the computer C output s, i.e. $$I_C(s) = \min_{C(p)=s} lg(p)$$
- Definition 3. A random binary string s is one having the property that $$I(s) \approx lg(s).$$
- Theorem 1. There is a constant c such that $I(s) \leq lg(s) + c$ for all s.
- Theorem 2. There are less than 2^n binary strings of complexity less than n.

A Kind of Berry Paradox

- Suppose you want to know whether a given string is random. Can you prove it to be so?
- Say you want to find "the first string that can be proven to be of complexity greater than 1000000000". There is always a program log(n+c) bits long that can calculate the first string that can be proven to be of complexity greater than n (a proof checker).

A Kind of Berry Paradox (II)

- Given this program and large n the test code´s length log(n+c) will be less than n.
- It is absurd for a string not to have a program of length n *and* to have one (vis. the test code). Such a string cannot exist!
- So: For all sufficiently great values of n it cannot be proven that a *particular* string is of complexity greater than n.
- Program-size complexity is *uncomputable*!

Incompleteness

Given this result we arrive at a sort of incompleteness: for a formal system with n+c bits of axioms it is possible to determine each string of complexity less than n and the complexity of each of these strings, and it is possible to exhibit each string of complexity equal or greater than n, *without* being able to know by *how much* the complexity of each of these strings exceeds n.

Algorithmic Probability

AIT introduces a 2nd complexity measure that includes all programs which compute a string s.That is the probability that a program the binary code of which is produced by coin tossing generates s. It is:

$$P(s) = \Sigma_{C(p)=s}\ 2^{-|p|}$$

i.e. each program of length k producing s adds 2 to the minus k to the algorithmic probability of s.

Further Concepts

- Given algorithmic probability further properties of it and of algorithmic information content can be investigated in AIT, for example: relative complexity of two strings, mutual complexity, algortihmic independence, and so on.
- We won´t go into the details.
  The basic idea here is that of algorithmic information content.

Sources

- Chaitin has written several books and lots of papers on AIT (many of them with the same content), most of which are available online (http://www.umcs.maine.edu/~chaitin/). See, for example:
- *The Unknowable.* [a popular overview]
- *Algorithmic Information Theory*. 1997[3].
- *Information, Randomness and Incompleteness*. 1997[2]. [a collection of his papers]