

## Re: an true information theory

**Source:** <http://sci.tech-archive.net/Archive/sci.math/2005-03/1364.html>

---

**From:** Stephen Harris (cyberguard1048-usenet\_at\_yahoo.com)

**Date:** 03/03/05

Date: Thu, 03 Mar 2005 22:27:15 GMT

"Daryl McCullough" <stevendaryl3016@yahoo.com> wrote in message news:d078vh0rrs@drn.newsguy.com...

> *Eray says:*

>

>>"Can the rule finding itself be done by some computer program?"

>>Interestingly, it is mathematically proven that there can be no

>>computer program which can eventually find (synonym: learn) these

>>(algorithmic) rules for all sequences which have such rules! "

>>

>>Here, the exclamation mark at the end of the first paragraph suggests

>>that the answer is a NO.

>

> He doesn't just *\*suggest\** that the answer is no, he is stating

> it. However, he is just saying that there is no such thing as

> a *\*perfect\** pattern recognizer program. That isn't saying anything

> about the impossibility of AI, because humans aren't perfect

> pattern recognizers, either.

>

>>That the rule finding itself cannot be done by

>>any program. However, he cannot come out and say it,

>>because that would not sound "scientific".

>

> What's unscientific about it? It's a provable fact about

> algorithms: There is no algorithm P which can reliably

> take the outputs of a second program Q and figure out

> the code for Q from its outputs.

>

> More precisely, let's have three notions of "figuring out

> an algorithm":

>

> 1. P can only make one "guess". Given an infinite sequence

> of outputs from Q, program P can in a finite amount of time

> halt and return the code for Q.

>

> That's obviously impossible, because if P stops after seeing n

> outputs, and figures out that the algorithm must be A1, there

> is always the possibility that the real algorithm is A2, where

- > *A2 produces identical outputs to A1 for the first n steps, and*
- > *different outputs thereafter.*
- >
- > *2. P can make as many guesses as it likes, but may only*
- > *change its guess if the guess is proven wrong. (That is,*
- > *at stage n, P makes a guess A for the algorithm. If at*
- > *a later stage, program Q makes an output that is inconsistent*
- > *with algorithm A, then P may change its guess).*
- >
- > *It's a little harder to see, but if P follows such a rule, it*
- > *can't be guaranteed to eventually figure out the correct algorithm.*
- > *The reason for this is the unsolvability of the halting problem.*
- > *Suppose that P's current guess is algorithm A, and P sees outputs*
- > *o\_0, o\_1, o\_2, ... o\_n. P knows that algorithm A produces outputs*
- > *o\_0, o\_1, ... o\_{n-1}, but P hasn't run A long enough to know whether*
- > *A's next output is o\_n or not. Then how long does P wait to decide*
- > *whether output o\_n is consistent with the algorithm being A? P*
- > *has no way of knowing whether A might eventually output o\_n, so*
- > *he has no way of knowing whether A is consistent with output o\_n.*
- >
- > *A third notion of "correctly guessing the algorithm" is yet more*
- > *complicated:*
- >
- > *3. P can make as many guesses as it likes, and may change its*
- > *guess at any time (and can change back to a previous guess, if*
- > *it wants). We will say that P is "infinitely often" correct if*
- > *there is an infinite number of stages n such that P's guess at*
- > *stage n is correct and no other wrong guess appears infinitely*
- > *often. That is, P makes an infinite sequence of*
- > *guesses for the algorithm: A\_1, A\_2, ... where A\_n and A\_m may*
- > *be equal, even when n is not equal to m. From this infinite*
- > *sequence of algorithms, strike off every algorithm that appears*
- > *only a finite number of times. If the resulting list has only*
- > *one algorithm in it, and that algorithm is correct, then P*
- > *has guessed correctly.*
- >
- > *I think with this very weak sense of "guessing the algorithm" it is*
- > *possible for P to correctly guess every algorithm.*
- >
- > --
- > *Daryl McCullough*
- > *Ithaca, NY*
- >

I appreciate your comments, but I have to question them.

- > *From this infinite*
- > *sequence of algorithms, strike off every algorithm that appears*
- > *only a finite number of times. If the resulting list has only*
- > *one algorithm in it, and that algorithm is correct, then P*
- > *has guessed correctly.*

Isn't this description equivalent to suggesting that from the infinite sequence of digits in the expansion of Pi, that we could determine there are no more 7's, thus only a finite number of 7's, thus Pi is does not pass randomness tests? As I understand it, no finite sequence can be proven random, and no individual infinite sequence can be proven truly random.

Unless one changes the definition of infinite, the algorithmic process to determine how many (finite) times a particular algorithm appears in an unending collection of algorithms is uncomputable.

This more than just a quibble. I've been reading about AIXI, which is Marcus Hutter's idea, <http://www.idsia.ch/~marcus/ai/index.htm>

Abstract: "Decision theory formally solves the problem of rational agents in uncertain worlds if the true environmental probability distribution is known. Solomonoff's theory of universal induction formally solves the problem of sequence prediction for unknown distribution. We unify both theories and give strong arguments that the resulting universal AIXI model behaves optimally in any computable environment. The major drawback of the AIXI model is that it is uncomputable. To overcome this problem, we construct a modified algorithm  $AIXI^{t,l}$ , which is still superior to any other time  $t$  and space

$l$  bounded agent. The computation time of  $AIXI^{t,l}$  is of the order  $t \times 2^l$ ." [SH: On page 5 "Time Bounds and Effectiveness"]

"Non-effectiveness of  $AI@$ :  $@$  is not a computable but only an enumerable semimeasure. Hence, the output  $y_k$  of the  $AI@$  model is only asymptotically computable.  $AI@$  yields an algorithm that produces a sequence of trial outputs eventually converging to the correct output  $y_k$ , but one can never be sure whether one has already reached it. Besides this, convergence is extremely slow, so this type of asymptotic computability is of no direct (practical) use. Furthermore, the replacement of  $@$  by time-limited versions [LV91, LV97], which is suitable for sequence prediction, has been shown to fail for the  $AI@$  model [Hut00b]. This leads to the issues addressed next."

SH: AIXI is uncomputable and this seems to me what you have described in your third notion 3.

However,  $AIXI^{t,l}$  is computable, but intractable. I think "intractability" is a better mathematical concept to explain the limit of sequence prediction discussed by Case. This eliminates Eray's use of Godel's Incompleteness to misunderstand Case's essay and somehow construe Case's statement as the author supporting Penrose and Lucas arguments against strong AI using Godel's Incompleteness Theorem.

Minsky is his paper Neat vs. Scruffy says we cannot arrive at intelligence by running computer simulation programs of evolution. Over a couple of billion years there are a huge number of random factors that render such a task combinatorially explosive. What the program would be looking for is a rule of causal events which produce intelligence through evolution.

If an oracle were to provide such a rule, then we could build intelligence in a toy universe by following the sequence of events that actually created intelligence on earth; the oracle has filled in the random blanks. Finding such a rule is an example of discovery a rule in sequence prediction illustrated by another simple example by Case. The limitation is imposed by lack of time and computational resource constraints, but there is a procedure to determine an individual result in contrast to the in principle inability to resolve individual infinite random numbers.

I've read that quantum computing may improve tractability but that quantum computing won't solve the halting problem so that the concepts don't seem to have the same meaning. I think Case's example applies to finitely complex sequence prediction rules.

IOW, Case is talking about a complex sequence which is a priori known to have a rule. But this rule may not be discovered within polynomial time.

Undecidability, I think, applies when it is not known if there is a rule or not and there is no systematic way of determining if there is such a rule.

>> *"Can the rule finding itself be done by some computer program?  
>> Interestingly, it is mathematically proven that there can be no  
>> computer program which can eventually find (synonym: learn) these  
>> (algorithmic) rules for all sequences which have such rules! "  
>>*

Doesn't this mean that you can have 10 sequences which have rules and one can't find out what the rule is.

Isn't that different from having 10 sequences and you don't know whether they have rules or not, nor what the rule is.

The knowledge seems to have different levels of depth. Distinguishing 10 sequences which have rules at an individual level (known a priori) from 10 sequences selected randomly, which may or may not have rules seems to revert to the problem of selecting/discovering an individual infinite truly random number.

Known a priori sequences which are intractable are known to be finite (but very long) as in the class of examples Case is using.

sci.math: Re: an true information theory

Undecidability has the additional lack of information so that the sequences may be finite or infinite.

I can't see how Godelian Inc. has anything to do with Case's point.