On Random and Hard-to-Describe Numbers Seminar Talk

Tabitha Wan

29 April 2024

1 Berry's Paradox and the Un-provability of Randomness

- We can take the number: 1,101,121 = "one million one hundred one thousand one hundred twenty-one" → which is ten written words. We can also denote the same number as "the first number not nameable in under ten words" → which is nine written words
  - This creates an inconsistency for 1,101,121
- This is self-referential paradox or Berry's paradox who say that nameability and definability are too vague to be used without restriction. There is this thing called systematic ambiguity where words like nameable, satisfiable, true, false, function, property, class, relation, cardinal, and ordinal give rise to vicious circle fallacies.
  - Take $N(x)$ = the number of words to name x can be seen as ill-defined for all but a finite x
  - Instead take $C(x)$ which can be the output of an algorithm: $C(x)$ = the number of bits in the smallest program to compute x. If we have binary integer p is a program to compute x when a standard universal computer has input p and only computes x and then halts. We can see that P describes x and such $C(x)$ is well-defined
  - We can not directly compute $C(x)$ if we want to avoid Berry's paradox. If we were to compute$C(x)$ directly we would be able to create a program q which would compute the least x for which $C(x)$ exceeded the number of bits in q.
- There is also Martin Gardner's paradox on classifying natural numbers as interesting or dull. We can not do so because all numbers would be interesting on account that the first few numbers would be interesting. Essentially if there non-empty set of uninteresting natural numbers, there would be a smallest uninteresting number – but the smallest uninteresting

number is itself interesting because it is the smallest uninteresting number, thus producing a contradiction.

  - Similarly, we could define an interesting number (that has no paradox) as one computable by a program with fewer bits than the number itself. The dull or random number would be algorithmically incompressible. Also, most numbers would be dull or random.

- By that definition of randomness, Chatain demonstrated a fact from Godel's Incompleteness Theorem:

  - Although most numbers are random only finitely many of them can be proved random given a consistent axiomatic system. All systems whose axioms and rules of inference that require n-bits to describe can not prove the randomness of a number longer than n-bits.
  - In general, Godel's Incompleteness Theorems are concerned with the limits of provability in formal axiomatic theories

2 The Search for a "Random" Real Number

- Many people have conjectured at irrational numbers are random and such are normal because every digit and each block of digits of any length occurs with the same frequency. We can easily see that no rational number is normal in any base because the digit sequences of rational numbers are eventually periodic.

- The same can not be easily shown for an irrational number, take a look at

$$C = 0.12345678910111213141516171819202122232425262728 29...$$

  the D.G. Champernowne consists of numbers written in increasing order... We do not know if these numbers are normal to every base.

- π may be random in the sense of being normal though it is not unpredictable similar to Champernowne's number. Think about the fact that we have computed 31 trillion digits of it. We question if there is a sequence that is so random that there can be no computeable betting strategy that you can have infinite gain. We know that any random number is normal to every base and due to probability theory we can say that almost all real numbers are random in this strong sense but what about a specific random number?

  - No specifically definable real number can be random due to the fact there are uncountably many real numbers but a countable number of definitions therefore the number must be uncomputable. If it was computable then you would have the perfect betting strategy.

- The uncomputable number may be described as k by the Halting Problem
  - The halting problem is a classic unsolvable problem in computing theory. It is the problem of distinguishing programs that come to a spontaneous halt from those that run on indefinitely.
  - It may look solvable because if a program halts then we can see this by running the program long enough and there are programs that can be easily proven to halt or not halt without running the program. The problem is not finding the particular cases but solving the problem in general. It can be shown that there is no effective prescription for deciding the long to run a program that waits long enough to reveal the halting of all halting programs, nor is there a consistent system of axioms strong enough to prove the non-halting of all non-halting ones. The un-solvability of the halting problem is derived from and equivalent to the fact that most random integers can't be proved random.
  - One such result shows that the halting problem is undecidable: no computer program can correctly determine, given any program P as input, whether P eventually halts when run with a particular given input. Kleene showed that the existence of a complete effective system of arithmetic with certain consistency properties would force the halting problem to be decidable, a contradiction
- Better yet we can define Ω = the halting probability of a universal computer where the program is generated randomly by tossing a fair coin when the computer requests another bit of input.
  - We know that Ω is a real number between 0 and 1
  - Dependent on the language the probability may be closer to 0 than 1
  - It can be shown that the first few digits of Ω would look more random than Champernowne's number
- Ω has three related properties that make it unusual:
  1. It encodes the halting problem in a compact form.
  2. It is algorithmically incompressible as there exists a constant c such that the first n bits of Ω are never expressible as the output of a program smaller than n-c bits.
  3. No computeable gambling scheme can make infinite profit betting against it.
- Ω is much more compact than K such that knowing the program's first n bits can allow you to solve any program of up to n bits in length.

- Say we have program p that is n-bits. Then p has n coin tosses having a probability of 2^{-n} and if it halts than it contributes that amount of probability to total halting probability Ω. If Ω_n is the first n bits of Ω then $\Omega_n < \Omega \leq \Omega_n + 2^{-n}$.
  - We can decide the halting of p. We find an unending search for all programs that halt of any length until enough halting programs have been found to account for more than omega n of the total halting probability. Then p is part of the programs that halted so far or it will never halt.
- There are unproved conjectures about the non-existence of something that only need a counter-example to disprove the conjecture. These conjectures can typically be described by it being encoded in the halting of small programs, such only the first few thousand digits of Ω would be needed to solve these these "finitely refutable" conjectures. Some well known conjectures like π is normal or are is an infinite amount of twin primes are not decidable by any finite amount of direct evidence. One of the most important conjectures is that P or polynomial time does not equal non-deterministic polynomial time. To prove something like this you would need a lot of quantifiers that may in turn say if it will or will not halt (for all says that it will run infinitely) and although a high level conjecture like this will not be able to decided by Ω it may be described indirectly Ω. But something like the existence of twin primes could be solved by using Ω because one could look at $10^n and 10^{n+1}$ and it would be decided by the early digits of Ω because the nonhalting of a simple program would look for an very large gap in the distribution of twin primes. Though there are still some statements that can not be reduced to halting problems like the statement of Ω itself.
- There are a class of statements that can be described by Ω, these statements are like: proposition p is provable in axiomatic system A.
  - Axiomatic system is a set of primitive notions and axiosn to logically derive theorems and said to be consistent if it lacks contradiction.
  - Say proposition with system A take n-bits to describe then a program of about n bits will halt if and only if p is provable in A
  - By having the first few thousand bits of omega then we can determine if p is provable, refutable, or independent in A
  - It follows that Z is a Turing machine for which the question of whether or not it halts when run indefinitely is equivalent to the consistency of ZFC. It follows that just as ZFC cannot prove its own consistency (assuming ZFC is consistent), ZFC also cannot prove that Z will run forever. While the undecidability of the halting problem tells us that there cannot exist an algorithmic method for determining whether an arbitrary Turing machine loops or halts, Z is an example of a

specific Turing machine whose behavior cannot be proven one way or the other using the foundation of modern mathematics.

- Ω is random: incompressibility and impossibility of betting against it: It is a message that appears random but is totally informative because all redundancy has been squeezed out
- Why is Ω algorithmically incompressible?
  - Start by denoting k = 1, 2, 3, ... where each k represents stages
  - At each k stage, we can run every program up to k bits for k seconds
  - Then we can compute Ω_k) (the halting probability at stage k). We know that Ω_k) is a subset of all halting programs that halt eventually whereas Ω is based on all programs that halt therefore $\Omega_k) \leq$ omega
  - As k increases Ω_k) gets closer to omega such that Ω_k)'s first bits will get closer to Ω
  - As soon as the first N bits are correct then we know we have encountered every program up to N bits in size that will ever halt
  - Now we can use the first N bits of Ω to solve the hΩomega with a program less than N bits long. Then combine that program with the one carrying out the Ω_k) algorithm. The program will be shorter than N bits which solves the halting problem. We know that this program does not exist and such Ω is incompressible.
- Why can we describe no computable gambling scheme that can win an infinite profit betting against Ω?
  - Let G be a gambling schedule of g bits are able to multiply the gambler's initial capital and 2^k fold by betting on some N of initial bits of Ω. Say this program also makes single bets. We can suppose it is the same gambling scheme applied to other inputs besides omega. On most bets of N it would fail to acheieve the goal but on a few it would succeed. We may use G to enumerate all finite inputs on which G would quit successfully, the set has a 2^{-n} probability or less contributed at omega n.
  - It can be shown that about $n - k$ bits suffice to locate Ω_n) therefore omega n can be computed by a program of g + n - k bits and k must not be much greater than g without violating incompressibility of omega. No g-bit gambling scheme betting on omega can multiply the initial capital by more than 2^g which is the amount of knowing g bits of omega and betting only on those bits.