TURING ZETA FUNCTIONS AND PARTIAL RANDOMNESS

Mike Stay
email: mike@math.ucr.edu

(Work done at University of Auckland with Cristian Calude)

What is Information?

1686 - Year before Newton's *Princ*

- How to distinguish a world describable by science from one that isn't?

- "Lawful behavior"

- Data fitting

- Law is a <u>smaller</u> description of the data

What's a description?

1934 - Church's lambda calculus

Everything's a function

Numbers as functions: Church nume

$$n(f)(z) = f^n(z) = f(f(\cdots(z)))$$

$$\text{zero}(f)(z) = z$$
$$\text{one}(f)(z) = f(z)$$
$$\vdots$$
$$\text{inc}(n)(f) = f(n)$$

$$\begin{aligned}\text{inc}(\text{zero})(f)(z) &= f(\text{zero})(f)(z) \\ &= f(\text{zero}(f)(z)) \\ &= f(z) \\ &= \text{one}(f)(z)\end{aligned}$$

so

$$\text{inc}(\text{zero}) = \text{one}$$

All of arithmetic

If / then:

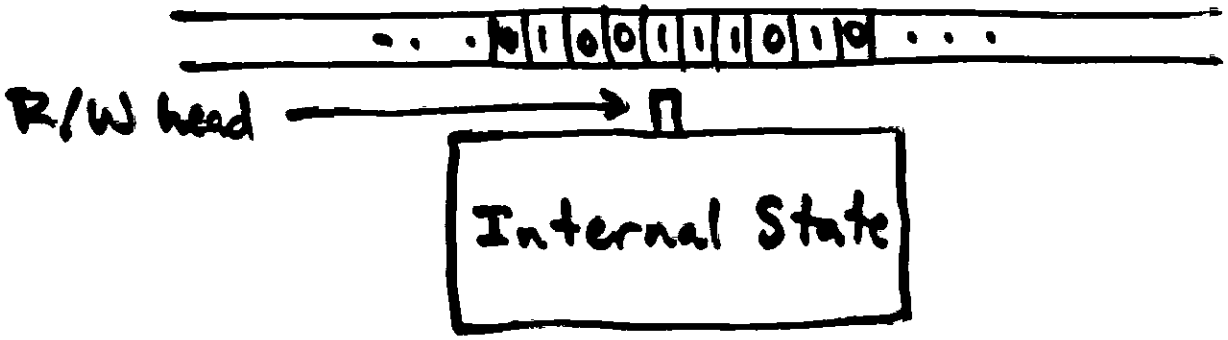
$$K(x)(y) = x$$
$$I(x) = x$$
$$P(x)(y)(z) = z(x)(y)$$

$$P(x)(y)(K) = K(x)(y) = x$$
$$P(x)(y)(K(I)) = K(I)(x)(y)$$
$$= I(y)$$
$$= y$$

Seems complicated because it's so simple But LISP, Haskell, Javascript, PERL all use the idea of functions of functions. The first two have little else: "Lambda calculus with syntactic sugar"

1934 Turing Machines

Infinite tape



Up to three possible operations at each step

- change bit
- change state
- move L/R

Turing showed they compute the same class of functions.

Formally, we say a Turing Machine is a partial computable function from strings to strings

$T: \{0,1\}^* \xrightarrow{\circ} \{0,1\}^*$

"partial", i.e. may not be defined

Universal Turing Machine

A TM U is universal if for any TM S, $\exists c_s$ s.t. $\forall x \; \exists x'$ s.t.

$$U(x') = S(x) \quad \text{and} \quad |x'| < |x| + c_s$$

A Universal machine can simulate any other machine with only a constant difference in the length of the input.

Domain of a Turing Machine

$\text{dom}(T)$ is that subset of $\{0,1\}^*$ on which T is defined (i.e. infinite loops and syntax errors are not in the domain)

Algorithmic Information Theory

"A description of data is a program whose output is that data."

(Relative to choice of programming language)

The *complexity* $H(x)$ of a string x is

$$H(x) = \min\{|p| : U(p) = x\}$$

The complexity is the length of the elegant program for x, where an elegant program is the shortest with that output.

Kolmogorov, Levin, Solomonoff, Chaitin

mid 1960's

FAS

- Axioms
- Rules

$\Rightarrow$ • Theorems

Theorem checker

- Does this theorem prove that the program P is elegant?

- Is $|P| > |FAS| + |Theorem\ checker|$?

- If yes to both, run U(P) and return the same result.

Contradiction or No such proofs for $|P| > N$!

Chaitin's Omega (1974)

$$\Omega = \sum_{p \in \mathrm{dom}(U)} 2^{-|p|}$$

Thm. Bits of Ω are random, i.e.

$$H(\Omega[m]) \geq m - c,$$

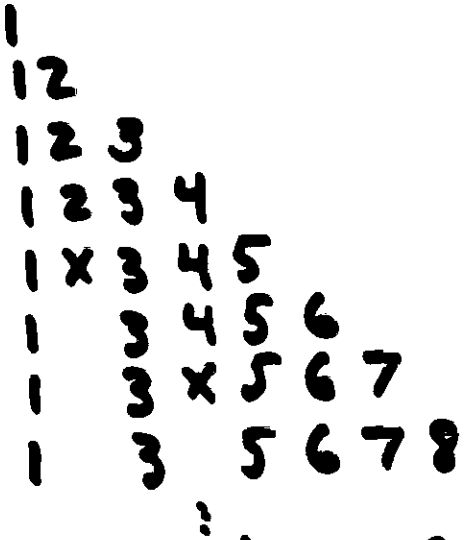
where $x[m]$ denotes ~~the m~~

$$\frac{\lfloor 2^m x \rfloor}{2^m}.$$

Proof.

Given Ω up to 2^{-m}, we know the halting status of all programs whose length is $\leq m$:

Run the programs in parallel



Add up contributions of halted programs.

When m bits match, no more p's can halt, because otherwise the contribution $2^{-|p|} > 2^{-m}$ and one of the bits would change!

There is a program $\Psi(\Omega[m])$ that computes the outputs of the halting programs and chooses a string not in that set. Call the string x. Then

$$H(\Psi(\Omega[m])) = H(x) > m$$

By universality of U,

$$H(\Psi) + H(\Omega[m]) \geq H(x) > m$$

so

$$H(\Omega[m]) \geq m - H(\Psi) \approx m - c$$

□

Given a FAS and a theorem checker that looks for proofs about the value of the first m bits of omega, we can find its length $N = |\text{FAS}| + |\text{Theorem Checker}|$

This program cannot prove the values of more than $m = N + c$ bits! So ZFC can't tell you more than a little bit of Ω.

Add k bits' worth of axioms and rules to ZFC. Will be able to prove at most k more bits of Ω.

The bits of Ω are irreducible informa

Given ~ 100000 bits of Ω for C or LISP or some such, could easily solve (in principle) all the Millenium Problems, many harder ones.

If you believe, like Wolfram or Fredkin, that the Universe is a digital computation on a Turing machine, then Omega becomes Borges' Aleph.

2002 Tadaki

Generalized Halting Probability

$$\Omega(s) = \sum_{p \in dom(U)} 2^{-s|p|}$$

For computable real $s>1$, $\Omega(s)$ is

$\frac{1}{s}$ - random, i.e.

$$H(\Omega(s)[m]) \geq \frac{m}{s} - c$$

2005 Stay

Zeta function of a Turing Machine

Let $bin(n):\mathbb{N} \to \{0,1\}^*$ be the binary expansion of n without the leading 1:

n_{10}	n_2	$bin(n)$
1	1	-
2	10	0
3	11	1
4	100	00
5	101	01
6	110	10
7	111	11
8	1000	000
⋮	⋮	⋮

The zeta function of a Turing machine T

$$\zeta_T(s) = \sum_{bin(n) \in dom(T)} n^{-s}$$

For Universal U, computable $s>1$

$$H(\zeta_U(s)[m]) \geq \frac{m}{s} - c$$

$K(x)$ is the complexity of x *given the length of x for free.*

Thm There are no random sequences with respect to K.

Pf. Assume $\exists$ binary sequence x such that

$$K(x[m]) \geq m - c.$$

Take c bits of x and interpret them as a length L. Then $K(x[c+1+L])$ is at most L, since the first $c+1$ bits don't give any new information: we could compute them from the last L bits. But $L < m - c = L + c + 1 - 2 = L + 1$.

□

Thm(Stay) There exist $\frac{1}{2}$-K-random sequences for all computable real $s>1$.

Proof follows from Chaitin's. The problem with $s=1$ is that the zeta function for a universal Turing machine diverges when you get the length for free.

Let

$$\eta(x) = \min\{n : U(\text{bin}(n)) = x\}$$

This is a better complexity measure for ζ_U; it is the elegant program itself rather than its length.

Given $\zeta_0(1)[m]$, what are upper + lower bounds on its complexity?

$$2^{m-c} \le \eta(\zeta_U(1)[m]) \le 2^{m+c'} = \underbrace{\text{print}}_{c'-2}\ \text{"}\underbrace{\qquad\qquad}_{m\ (=\zeta_0(1)[m])}\text{"}$$

$$\text{So } \Delta\eta = 2^{m+c'} - 2^{m-c} = 2^m(2^{c'} - 2^{-c}) \quad c, c' \ge 1$$
$$> 2^m(2 - 2^{-c})$$
$$> 2^m$$

What are upper and lower bounds on $\zeta_U(1)$?

$$\zeta_0(1)[m] < \zeta_U(1) < \zeta_U(1)[m] + 2^{-m}$$

$$\text{So } \Delta\zeta = 2^{-m}$$

$\Delta\eta\Delta\xi > 1$

Heisenberg!

Suppose that some FAS could compute bits of $\zeta_u(1)$, but not in order — we've seen that's impossible.

Alas, this fails, too. If the FAS can prove infinitely many bits, then we just need to supply the ones in between. Once we get more than c bits from the FAS, we supply less than $m-c$ bits to produce $\zeta_u(1)[m]$, which violates the inequality above.

Mesl: Uncertainty implies Incompleteness